

# IO流

## 第2节 【输入输出流】

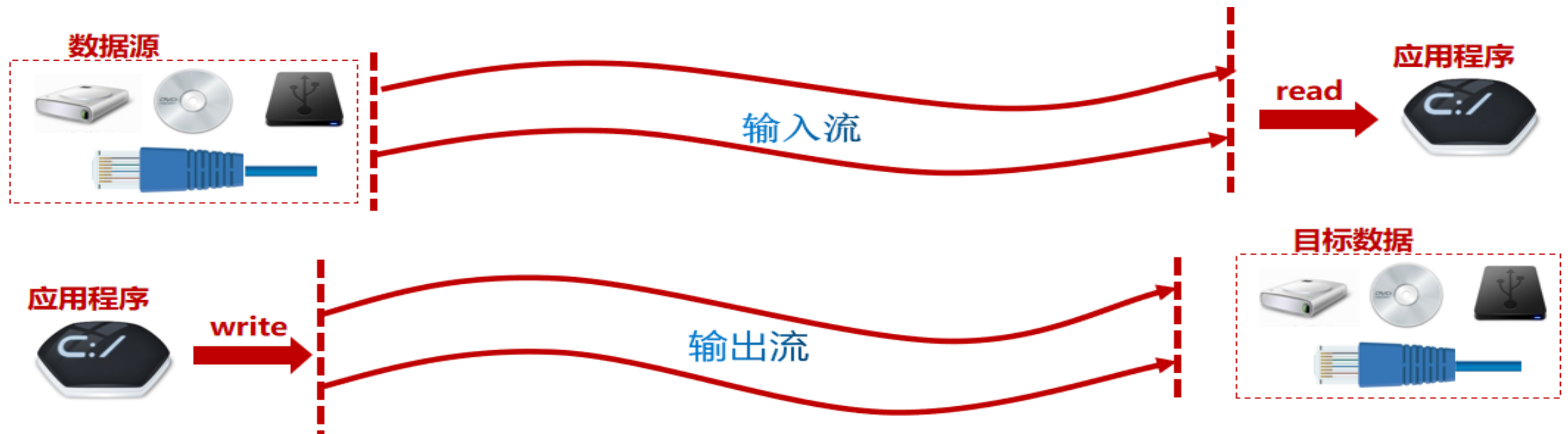


- 知识点1：IO流的概念与作用
- 知识点2：IO流的体系结构
- 知识点3：掌握字符的输入输出流
- 知识点4：掌握字节的输入输出流
- 知识点5：掌握System的标准输入输出流
- 知识点6：掌握字节字符转换流
- 知识点7：掌握随机访问文件流
- 知识点8：掌握字节数组流

# 知识点1: IO流的概念与作用

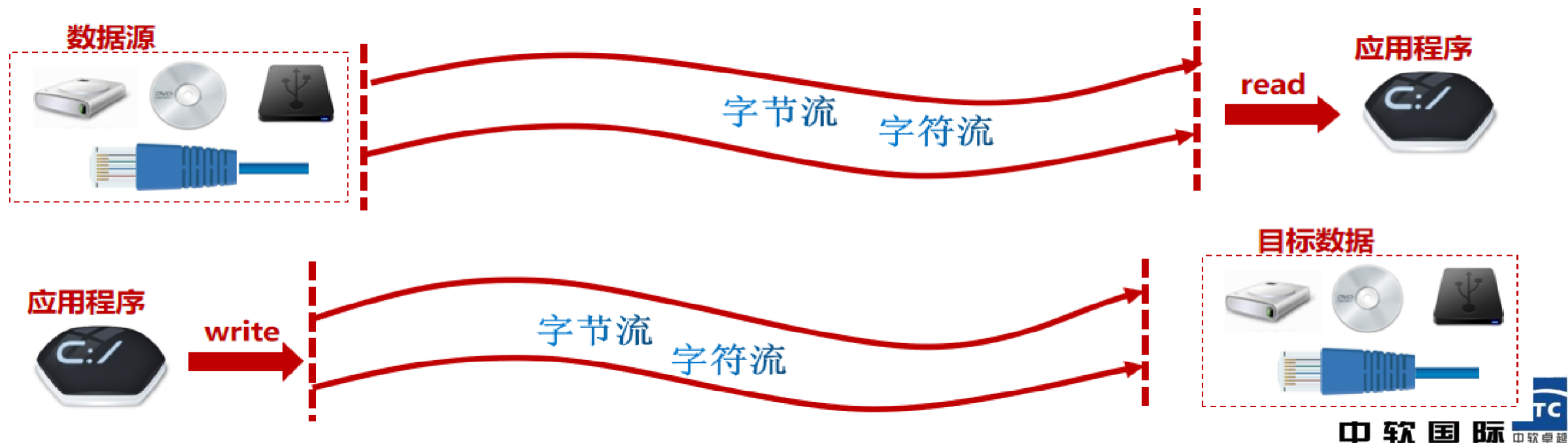


- IO流是什么:
  - I (Input) 输入, O (Out) 输出
- IO流的作用:
  - 数据在各个设备之间的传输, 是通过流的方式完成的
- 根据流动方向的不同, 流分为输入流和输出流



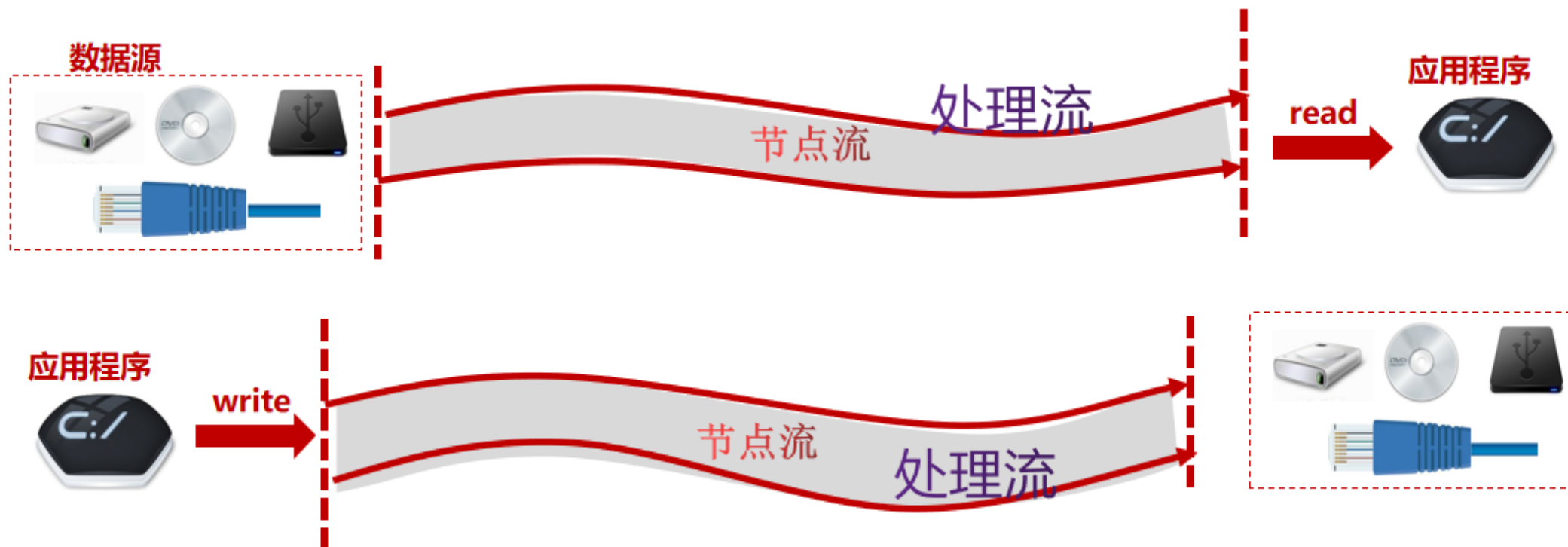
## 知识点1: IO流的概念与作用

- 根据流的格式不同，流分为字节流和字符流
- 程序中的输入和输出都是以流的形式保存的，流中保存的实际上全是字节文件。
- 所有文件的存储都是字节 (byte) 来存储，在磁盘上保留的并不是文件的字符，而是先把字符编码成字节，再存储这些字节到硬盘上，在读取时也是一个一个的读取以形成序列



## 知识点1: IO流的概念与作用

- 根据流的功能不同，又分为节点流和处理流
- 节点流：可以从某节点读数据或向某节点写数据的流
- 处理流：对已存在的流的连接和封装，实现更为丰富的流数据处理，提高流读写效率



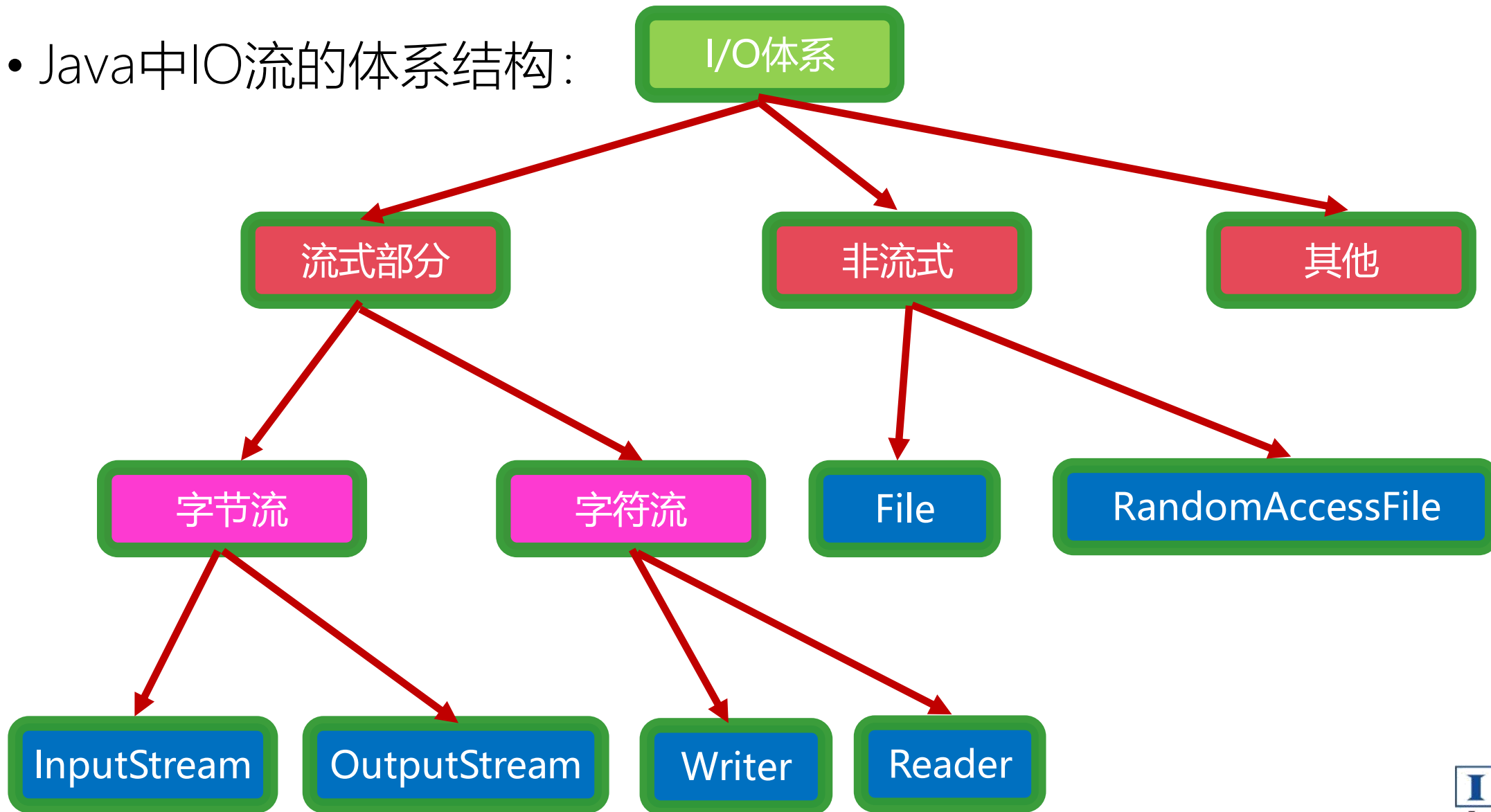
## 知识点2： IO流的体系结构



- 字节流的抽象父类
- InputStream字节输入流, OutputStream字节输出流
- 字符流的抽象父类:
  - Reader 字符输入流, Writer字符输出流
  - 注：由这四个类派生出来的子类名称都是以其父类名作为子类名的后缀。
  - 如：InputStream的子类FileInputStream。
  - 如：Reader的子类FileReader。

## 知识点2：IO流的体系结构

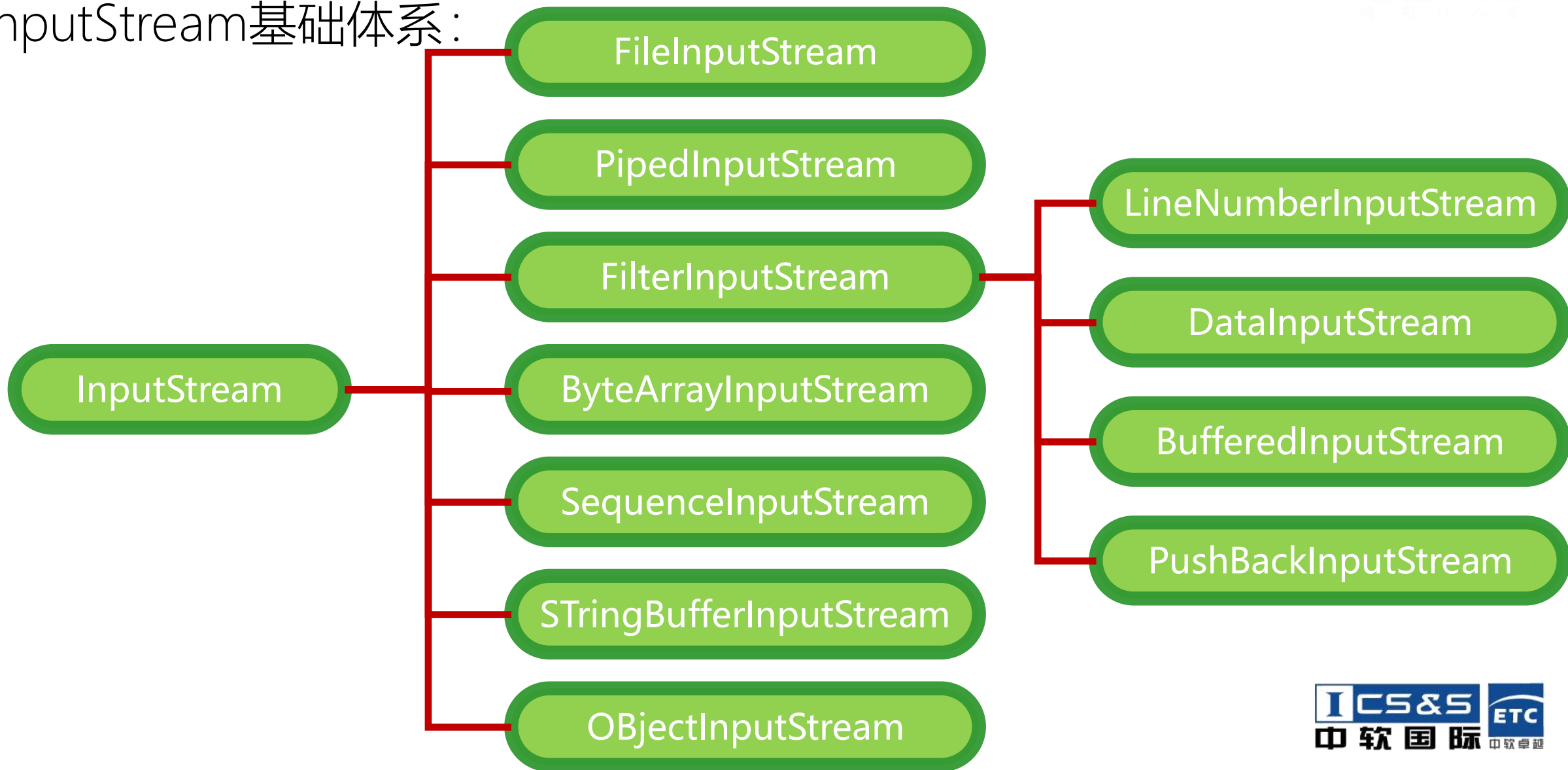
- Java中IO流的体系结构：



## 知识点2：IO流的体系结构



- InputStream基础体系：

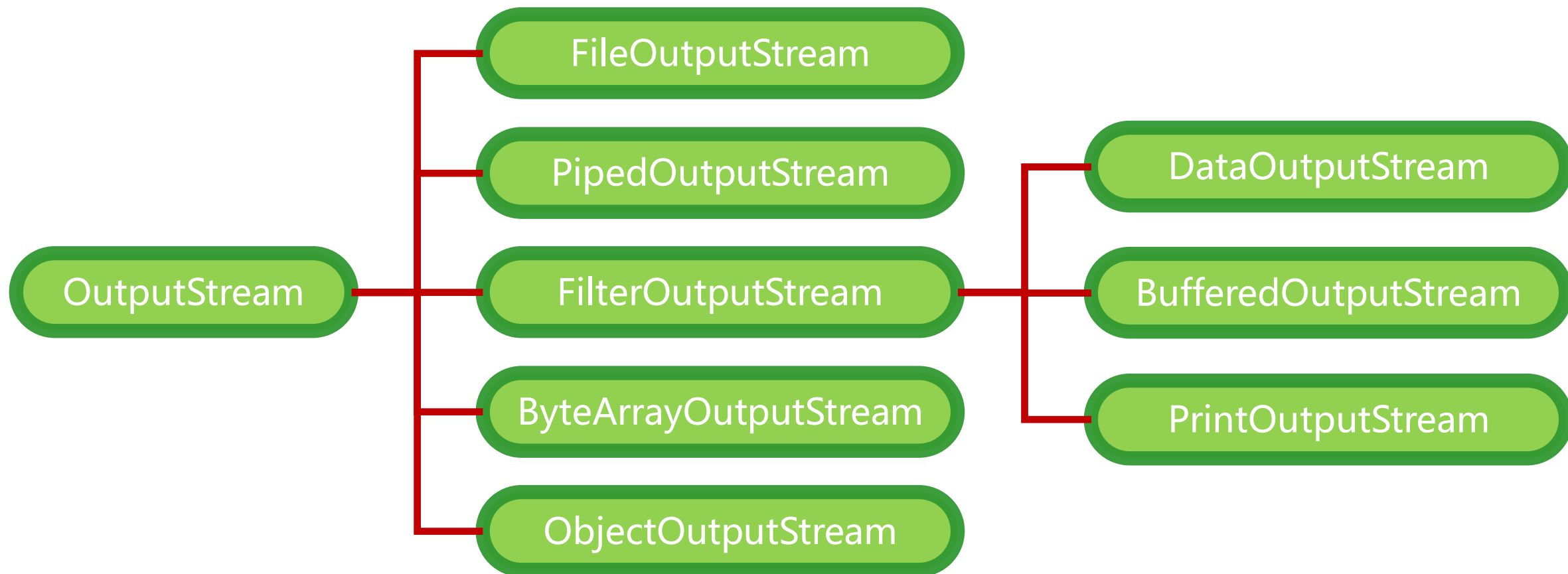




## 知识点2：IO流的体系结构



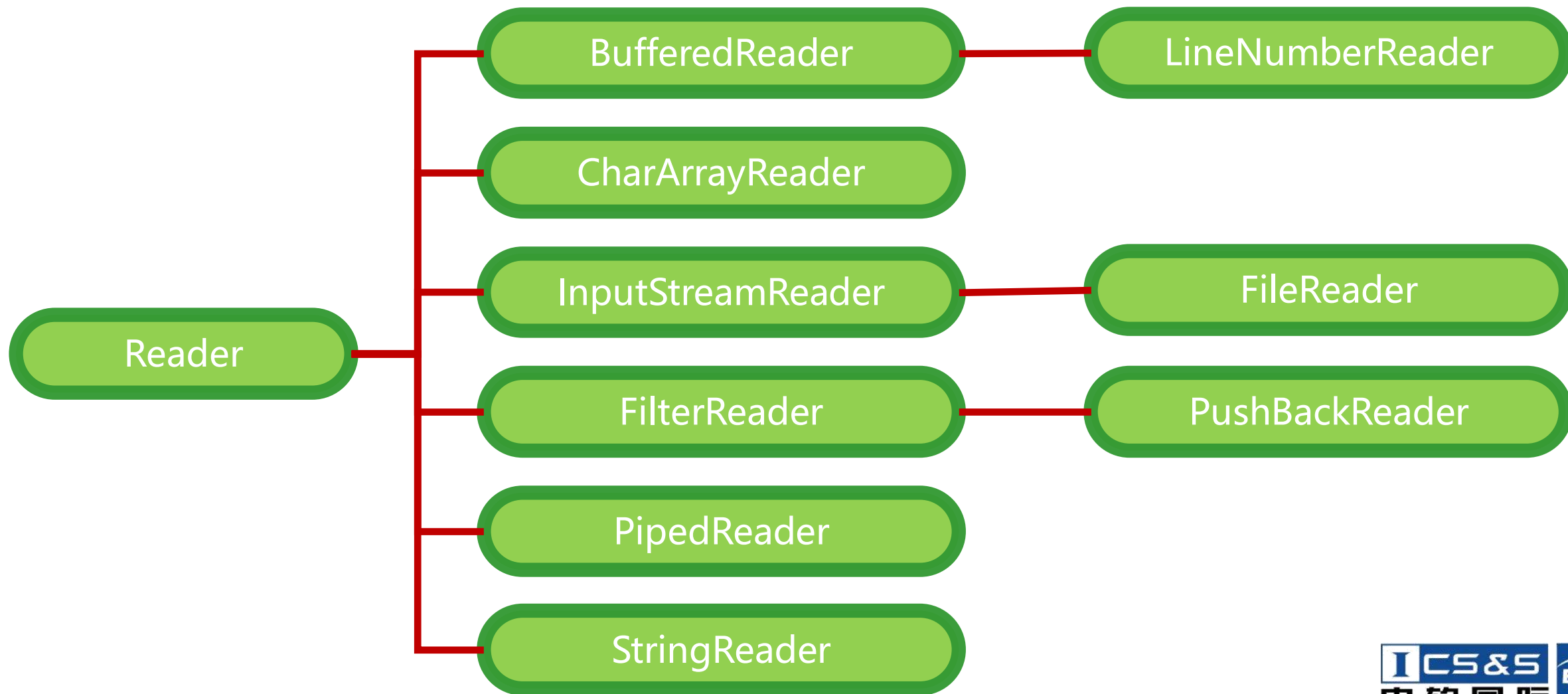
- OutputStream基础体系：



## 知识点2：IO流的体系结构



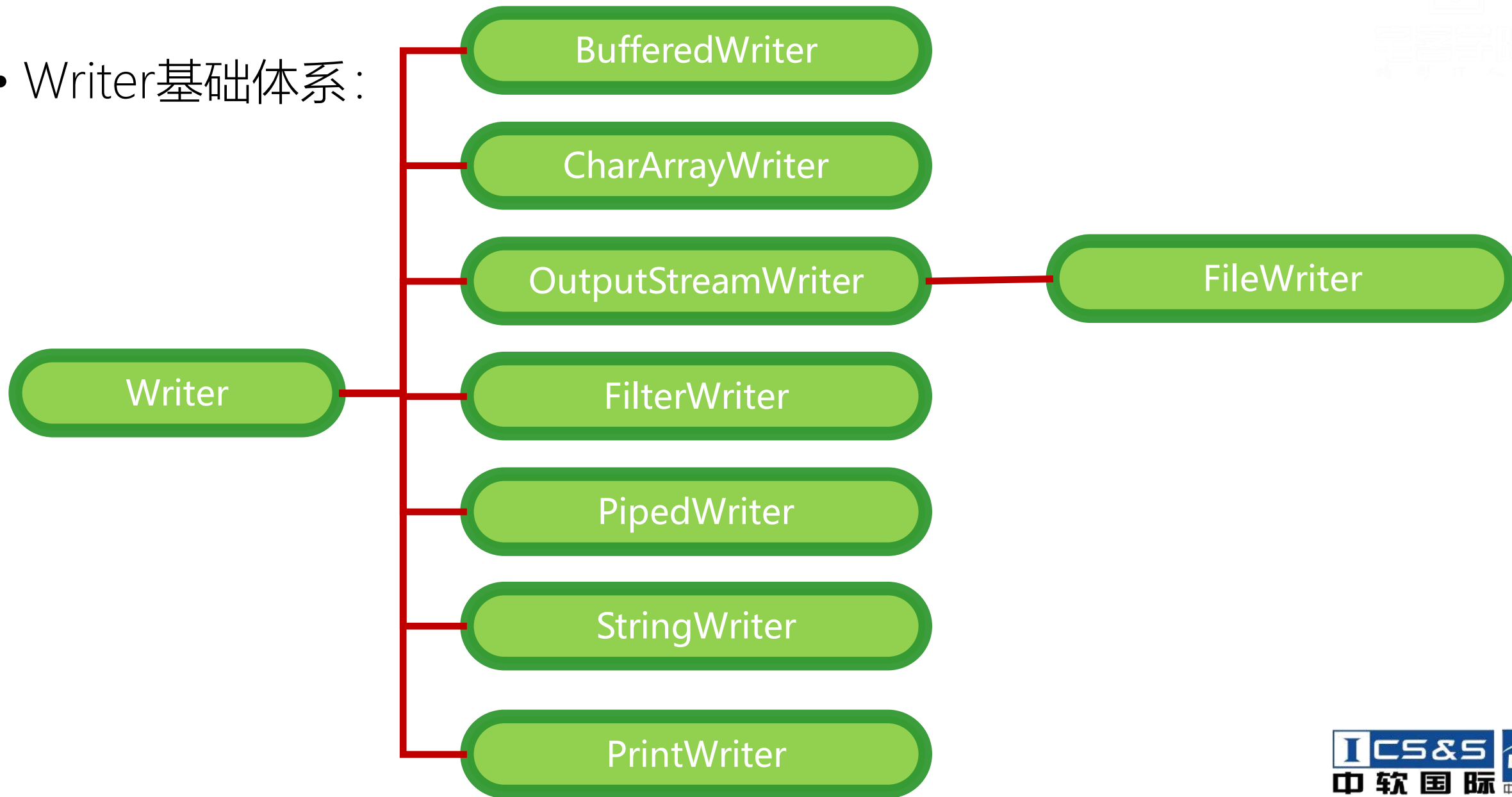
- Reader基础体系：



## 知识点2： IO流的体系结构



- Writer基础体系：



## 知识点3：掌握字符的输入输出流



- Writer抽象类里的方法

方法名	说明
abstract void close()	关闭此流，但要先刷新它
abstract void flush()	刷新该流的缓冲 字符流一定要刷新
void write(char[] cbuf)	写入字符数组，调用write(cbuf,0,cbuf.length)
void write(int c)	写入单个字符 如查写入的是97,则显示出来是a
void write(String str, int off, int len)	写入字符串的某一部分
void write(String str)	写入字符串,调用write(str,0,str.length())

- 由于Writer是抽象类不能创建对象，所以用它子类来完成写入字节的操作

FileWriter

BufferedWriter

## 知识点3：掌握字符的输入输出流

- FileWriter类是Writer的子类，是文件写入流，以字符流的形式对文件进行写操作，其构造方法有5种重载，以下是常用的几种：

构造方法	说 明
FileWriter(File file) throws IOException	使用File对象创建文件写入流对象，如果文件打开失败，将抛出异常，必须捕捉 如果文件存在，也会被覆盖
FileWriter(File file, boolean append) throws IOException	使用File对象创建文件写入流对象，并由参数append指定是否追加，异常情况同上
FileWriter(String name) throws IOException	直接使用文件名或路径创建文件写入流对象如果文件存在，也会被覆盖
FileWriter(String name, boolean append) throws IOException	直接使用文件名或路径创建文件写入流对象，并由参数append指定是否追加，异常情况同上

## 知识点3：掌握字符的输入输出流

例10

字符流： java.io.FileWriter用来写入字符文件的便捷类

需求：将"abc"字符串写入硬盘文件中。

这一行执行，会自动创建一个文件，有则覆盖

创建一个字符输出流对象。

```
FileWriter fw = new FileWriter("D:\\Test.txt");
```

这一行执行，如果a不存在则报异常

```
FileWriter fw = new FileWriter("D:\\a\\Test.txt");
```

这一行执行，如果存在就不会覆盖

```
FileWriter fw = new FileWriter("D:\\a\\Test.txt", true);
```

调用流对象的写入方法，将数据写入流,其实写到临时存储缓冲区中

- fw.write("abc");

刷新该流的缓冲，直接到目的地

- fw.flush();

- 不写flush方法会有什么结果呢？

关闭流资源，在关闭流资源前先调用flush方法，并将流中的数据清空到文件中。

- fw.close();

- 不写close方法，效率会越来越低

- close()内部原理

## 知识点3：掌握字符的输入输出流

例11

- 换行符



张总，您好！：  
世界那么大，我想去看看！

- java中的转义符"`\r\n`":
- windows下的文本文件换行符:`\r\n`
- linux/unix下的文本文件换行符:`\n`
- Mac下的文本文件换行符:`\`
- 要用`System.getProperty("line.separator")`代替 固定格式的换行符
- 具备平台无关性
- 一次编写，到处运行
- 更保险

# 知识点3：掌握字符的输入输出流

- Reader抽象类里的方法

方 法	说 明
public int read() throws IOException	读一个字符，返回0到65535（ 0x00-0xffff ）范围内的整数，如果已经达到流的末尾， <b>则为-1</b> 。
public int read(char[] cbuf) throws IOExceptionn	将字符读入数组，返回读取的字符数，如果已经达到流的结尾， <b>则为-1</b>
public abstract int read(char[] cbuf, int off, int len)throws IOException	将字符读入数组的一部分
public abstract void close() throws IOException	关闭的流

- 由于Writer是抽象类不能创建对象，所以用它子类来完成写入字节的操作

- 注意：read() 和read(char[])的区别

read(char [] ch,int off,int len) 什么时候用

BufferedReader

FileReader



## 知识点3：掌握字符的输入输出流



- FileReader类是Reader的子类，称为**文件读取流**，允许以字符流的形式对文件进行读操作，其构造方法有3种重载方式，以下是常用的几种：

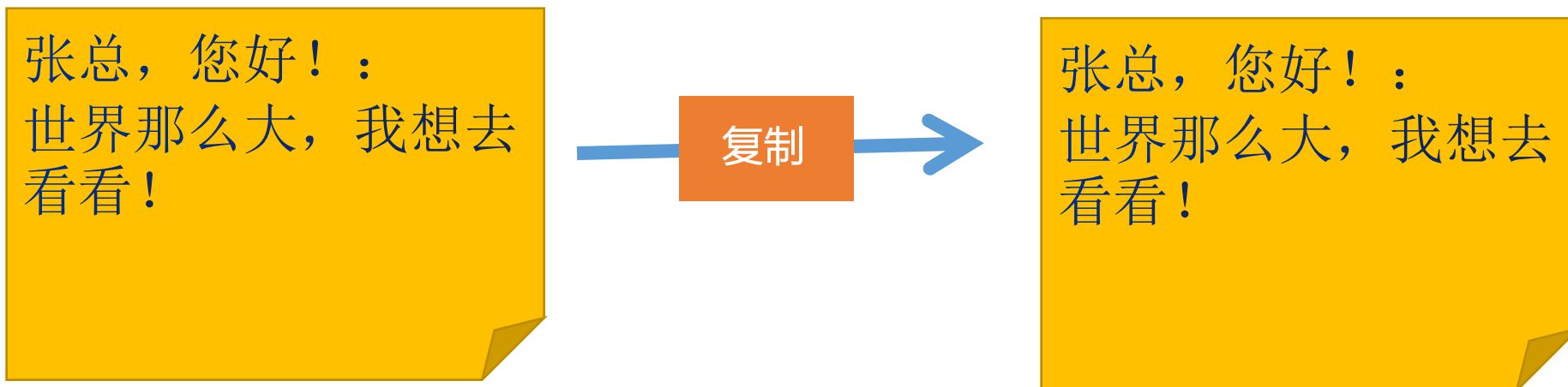
构造方法	说 明
FileReader(File file) throws FileNotFoundException	使用File对象创建文件读取流对象，如果文件打开失败，将抛出异常
FileReader(String name) throws FileNotFoundException	使用文件名或路径创建文件读取流对象，如果文件打开失败，将抛出异常

## 知识点3：掌握字符的输入输出流

- FileReader与FileWriter实现文件的复制

例13

宝善学院  
Bao Shan Academy



## 知识点3：掌握字符的输入输出流

例14

- BufferedWriter类里有一个方法要注意

方法名	说明
<code>public void newLine() throws IOException</code>	写一行行分隔符

- BufferedReader类里有一个方法要注意

例15

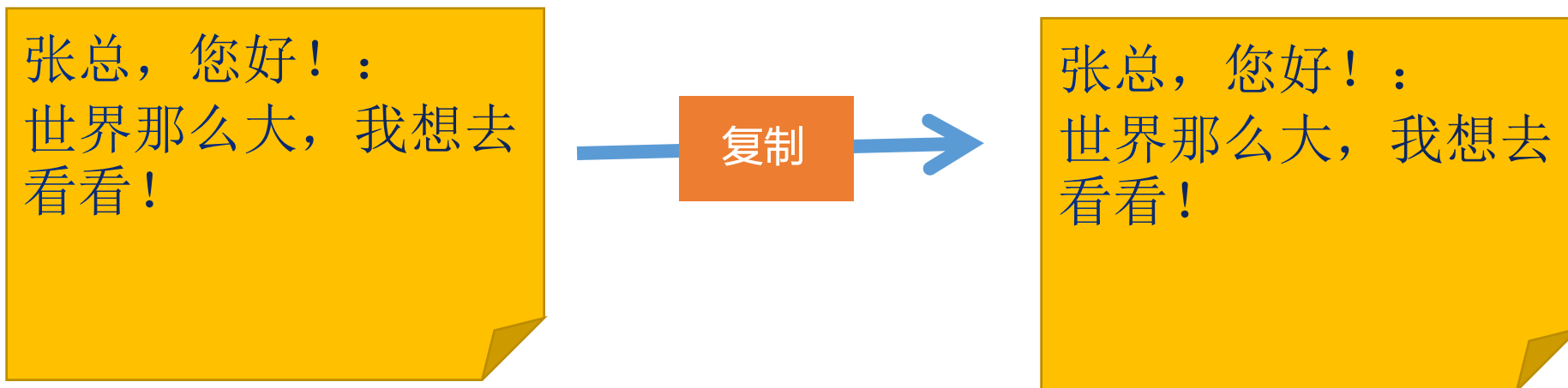
方法名	说明
<code>public String readLine() throws IOException</code>	读一行文字。一行被视为由换行符（'\ n'），回车符（'\ r'）中的任何一个或随后的换行符终止

- 说明：处理流和节点流的区别

## 知识点3：掌握字符的输入输出流

例16

- BufferedWriter BufferedWriter实现文件的复制



## 知识点3：掌握字节的输入输出流

例17

- OutputStream它不仅写入字节，字符，还可以写入图片等媒体文件

方法	说明
void close()	关闭此输出流并释放与此流有关的所有系统资源
void flush()	刷新此输出流并强制写出所有缓冲的输出字节
abstract void write(int b)	将指定的字节写入此输出流
void write(byte[] b)	将 b.length字节从指定的字节数组写入此文件输出流。

- 由于OutputStream是抽象类不能创建对象，所以用它子类来完成写入字节的操作

FileOutputStream

BufferedOutputStream

## 知识点3：掌握字节的输入输出流



- 向文件中写入字节，用OutputStream的子类java.io.FileOutputStream
- 类的构造方法有5种重载方式，以下是常用的几种

构造方法	说 明
<code>FileOutputStream(File file)</code> throws <code>FileNotFoundException</code>	使用File对象创建文件输出流对象，如果文件打开失败，将抛出异常
<code>FileOutputStream(File file, boolean append)</code> throws <code>FileNotFoundException</code>	使用File对象创建文件输出流对象，并由参数append指定是否追加文件内容，true为追加，false为不追加（将会直接删除以前的文件内容），异常情况同上
<code>FileOutputStream(String name)</code> throws <code>FileNotFoundException</code>	直接使用文件名或路径创建文件输出流对象，异常情况同上
<code>FileOutputStream(String name, boolean append)</code> throws <code>FileNotFoundException</code>	直接使用文件名或路径创建文件输出流对象，并由参数append指定是否追加，异常情况同上

## 知识点4：掌握字节的输入输出流



- java.io.FileOutputStream用来写入字节文件的便捷类。
- 需求：将"abc"字符串写入硬盘文件中。

创建一个字节输出流对象。

- `FileOutputStream fw = new FileOutputStream ("D:\\Test.txt");`

调用流对象的写入方法，将数据写入流,其实写到文件中

- `fw.write(数字);`
- `fw.write("abc".getBytes());`

关闭流资源`fw.close();`

- 关闭资源之前先判断下流
- 不写close方法，效率会越来越低
- 在finally中对流进行关闭

## 知识点4：掌握字节的输入输出流



- 用FileOutputStream将"abc"字符串写入硬盘文件中

```
FileOutputStream fs = null;
try {
    fs = new FileOutputStream("D:\\Test.txt");
    fs.write(1);
    fs.write("abc".getBytes());
} catch (Exception e) {

    e.printStackTrace();
} finally {

    if (fs != null) {
        try {
            fs.close();
        } catch (IOException e) {

            e.printStackTrace();
        }
    }
}
```

先判断



## 知识点4：掌握字节的输入输出流



- 用BufferedOutputStream向文件中写数据
- 注意：处理流不能单独使用要套在节点流的外面才可以使用

```
FileOutputStream fs = null;
BufferedOutputStream fos = null;
try {
    fs = new FileOutputStream("D:\\Test.txt");
    fos = new BufferedOutputStream(fs);
    fos.write(1);
    fos.write("abc".getBytes());
    fos.flush();
} catch (Exception e) {

    e.printStackTrace();
} finally {
    if (fs != null) { ← 判断
        try {
            fs.close();
        } catch (IOException e) {

            e.printStackTrace();
        }
    }
    if (fos != null) { ← 判断
        try {
            fos.close();
        } catch (IOException e) {

            e.printStackTrace();
        }
    }
}
```

## 知识点4：掌握字节的输入输出流

例18

- InputStream它不仅读取字节，字符，还可以读取图片等媒体文件

方法签名	说明
<code>void close()</code>	关闭此输出流并释放与此流有关的所有系统资源
<code>public int read() throws IOException</code>	从该输入流读取一个字节的的数据，如果达到文件的末尾，返回 -1
<code>public int read(byte[] b) throws IOException</code>	从该输入流读取最多b.length字节的数据到字节数组(返回读入的字节数)，文件的结尾已经到达，返回 -1
<code>public int read(byte[] b,                 int off,                 int len) throws IOException</code>	从该输入流读取最多len字节的数据为字节数组 (返回读出来的元素个数) b - 读取数据的缓冲区。 off - 目标数组 b的起始偏移量 len - 读取的最大字节数。

- 由于InputStream是抽象类不能创建对象，所以用它子类来完成写入字节的操作

FileInputStream

BufferedInputStream

## 知识点4：掌握字节的输入输出流



- java.io.FileInputStream用来写入字符文件的便捷类。
- 需求：将"abc"字符串从文件中读取出来。

创建一个字符输出流对象。

- `FileInputStream fis = new FileInputStream ("D:\\Test.txt");`

调用流对象的写入方法，将数据写入流,其实写到临时存储缓冲区中

- `fis.read();`
- `fis.read(byte的数组);`

关闭流资源`fw.close();`

- 不写close方法，效率会越来越低
- 在finally中对流进行关闭

## 知识点3：掌握字符的输入输出流

例19

- BufferedWriter BufferedWriter实现文件的复制



单曲 .MP3

歌手: The Flashbulb

所属专辑: Girls Suck E



纯音乐, 无歌词

复制



单曲 .MP3

歌手: The Flashbulb

所属专辑: Girls Suck E



纯音乐, 无歌词

## 知识点3：掌握字符的输入输出流



- BufferedOutputStream类里有一个方法要注意

方法名	说明
public void write(byte[] b) throws IOException	将 b.length字节从指定的字节数组写入此文件输出流。
public void write(byte[] b, int off, int len) throws IOException	b - 数据。 off - 数据中的起始偏移量。 len - 要写入的字节数。

- BufferedInputStream类里有一个方法要注意

方法名	说明
public int read(byte[] b) throws IOException	读入缓冲区的总字节数，如果没有更多的数据，因为文件的结尾已经到达， <b>-1</b> 。
public int read(byte[] b,int off, int len)throws IOException	读取到缓冲区的总字节数，如果没有更多的数据，因为文件的结尾已经到达， <b>-1</b> 。

- 说明：处理流和节点流的区别



## 知识点3：掌握字符的输入输出流

例20

- BufferedWriter BufferedWriter实现文件的复制



复制



复制



## 知识点5：System的标准输入输出流

- 程序对应的基本输入为键盘输入，基本输出为显示器输出。Java中，System类的in和out两个成员代表了基本输入输出的抽象
- System.in:
  - System.in是InputStream, 标准输入流, 默认可以从键盘输入读取字节数据
- System.out:
  - System.out是PrintStream, 标准输出流, 默认可以向Console中输出字符和字节数据
  - **注意：System.in流是没需关闭的**

## 知识点5：System的标准输入输出流



- InputStream类里的方法

构造方法	说 明
<code>public abstract int read() throws IOException</code>	从输入流读取数据的下一个字节，如果达到流的末尾， -1 。
<code>public int read(byte[] b) throws IOException</code>	从输入流读取一些字节数，并将它们存储到缓冲区b ，返回读取到缓冲区的总字节数，已经到达流的末尾，则是 -1 。

- PrintStream类有一些方法，这个类继承了OutputStream， OutputStream下面也有一些方法

构造方法	说 明
<code>public void print(char c)</code> 多类型参数	打印字符
<code>public void write(byte[] b) throws IOException</code>	将b.length字节从指定的字节数组写入此输出流
<code>public void write(byte[] b,int off, int len) throws IOException</code>	从指定的字节数组写入len字节，从偏移off开始输出到此输出流



## 知识点5：掌握System的标准输入输出流



- 读取键盘输入的数据并打印


```
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

System.out.println("请任意输入：");
String line = null;
try {
    line = br.readLine();
} catch (IOException e) {

    e.printStackTrace();
}finally{

    if(br != null){
        try {
            br.close();
        } catch (IOException e) {

            e.printStackTrace();
        }
    }
}
System.out.println("你输入是：" + line);
}
```



键盘输入

## 知识点5：掌握System的标准输入输出流



- 读取键盘输入的数据并打印

```
BufferedWriter bw = new BufferedWriter(  
    new OutputStreamWriter(System.out));  
  
try {  
    bw.write("1111");  
    bw.newLine();  
  
    bw.write("2222");  
    bw.newLine();  
  
    bw.write("3333");  
    bw.newLine();  
  
    bw.flush();  
} catch (IOException e) {  
    e.printStackTrace();  
}  
finally{  
    if(bw != null){  
        try {  
            bw.close();  
        } catch (IOException e) {  
            e.printStackTrace();  
        }  
    }  
}
```



输出到控制台

## 知识点6：掌握字节字符转换流



- 转换流：以将一个字节流转换为字符流，也可以将一个字符流转换为字节流
- 字节字符转换流位于java.io包
- OutputStreamWriter：可以将输出的字符流转换为字节流的输出形式。

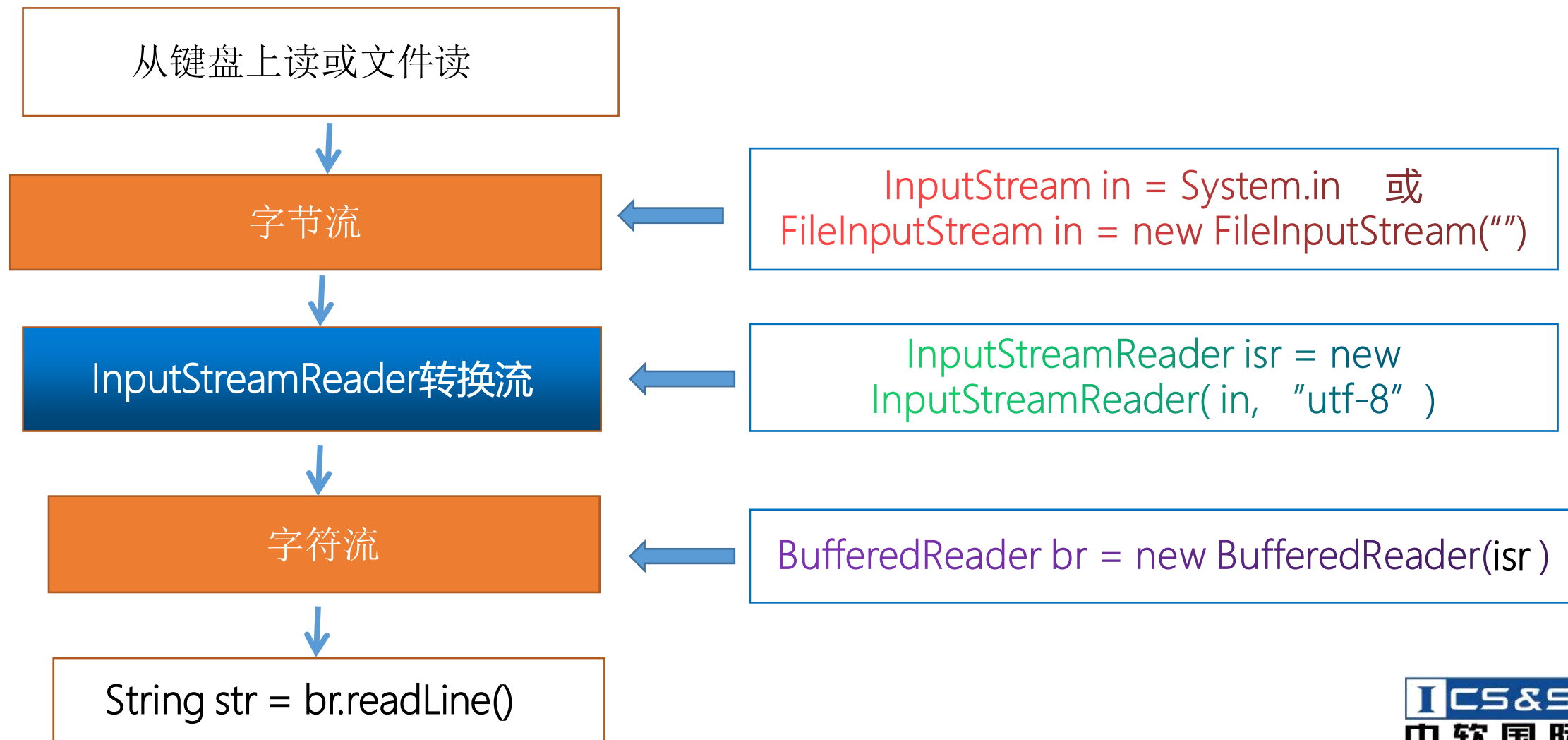
构造方法	说 明
<code>public OutputStreamWriter(OutputStream out)</code>	创建使用默认字符编码的 OutputStreamWriter。
<code>public OutputStreamWriter(OutputStream out, Charset cs)</code>	创建使用给定字符集的 OutputStreamWriter。

- InputStreamReader：将输入的字节流转换为字符流输入形式。

构造方法	说 明
<code>public InputStreamReader(InputStream in)</code>	创建一个使用默认字符集的 InputStreamReader
<code>public InputStreamReader(InputStream in, CharsetDecoder dec)</code>	创建使用给定字符集解码器的 InputStreamReader

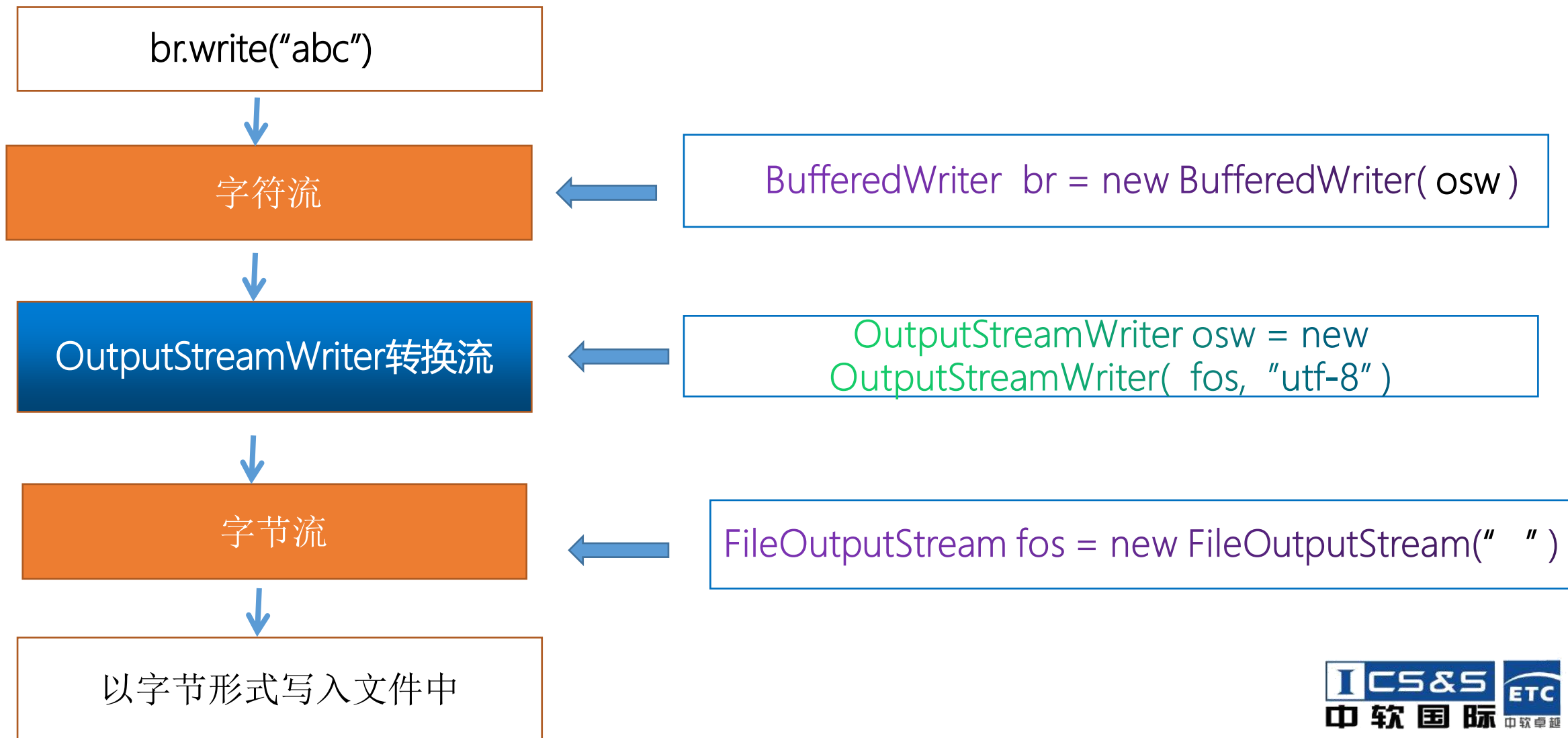
## 知识点6：掌握字节字符转换流

- 字节流转换成字符流图解



## 知识点6：掌握字节字符转换流

- 字符流转换成字节流图解



## 知识点6：掌握字节字符转换流



- 把字节流转换成字符流

```
try {  
  
    InputStream in = System.in;  
    InputStreamReader isr = new InputStreamReader(in);  
    BufferedReader br = new BufferedReader(isr);  
  
    FileOutputStream fos = new FileOutputStream("e:/aa.txt");  
    OutputStreamWriter osw = new OutputStreamWriter(fos, "utf-8");  
    BufferedWriter bw = new BufferedWriter(osw);  
  
    String str = "";  
    while ((str = br.readLine()) != null) {  
  
        if (str.equals("over")) {  
            break;  
        }  
        bw.write(str);  
        bw.newLine();  
        bw.flush();  
    }  
} catch (Exception e) {  
  
    e.printStackTrace();  
}
```

加上字符编码格式

## 知识点6：掌握字节字符转换流



- 把字节流转换成字符流简单写法

```
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
BufferedWriter bw = new BufferedWriter(new OutputStreamWriter(new FileOutputStream("e:/aa.txt"), "utf-8"))

String str = "";
while ((str = br.readLine()) != null) {

    if (str.equals("over")) {
        break;
    }
    bw.write(str);
    bw.newLine();
    bw.flush();
} catch (Exception e) {

    e.printStackTrace();
}
```

加上字符编码格式

## 知识点7：掌握随机访问文件流

- 随机流（RandomAccessFile）：此类的实例支持对随机访问文件的读取和写入
- 位于java.io包
- 特点
  - 该对象即能读也能写，一个对象就搞定
  - 该对象内部维护了一个大型 byte 数组，光标或索引在该数组任意位置读取或写入任意数据
  - 可以通过getFilePointer方法获得光标的位置和通过seek方法设置光标位置
  - 该对象将字节输入流和输出流进行了封装
  - 该对象源或目的，只能文件，通过下面的构造方法就可以看出

构造方法	说 明
public RandomAccessFile(File file, String mode) throws FileNotFoundException	File只能是文件（文本文件或者媒体文件） mode 的值可选 "r"：可读，"w"：可写，"rw"：可读性
public RandomAccessFile(String name, String mode) throws FileNotFoundException	String只能是文件（文本文件或者媒体文件） mode 的值可选 "r"：可读，"w"：可写，"rw"：可读性



## 知识点7：掌握随机访问文件流



- 随机流的方法有很多

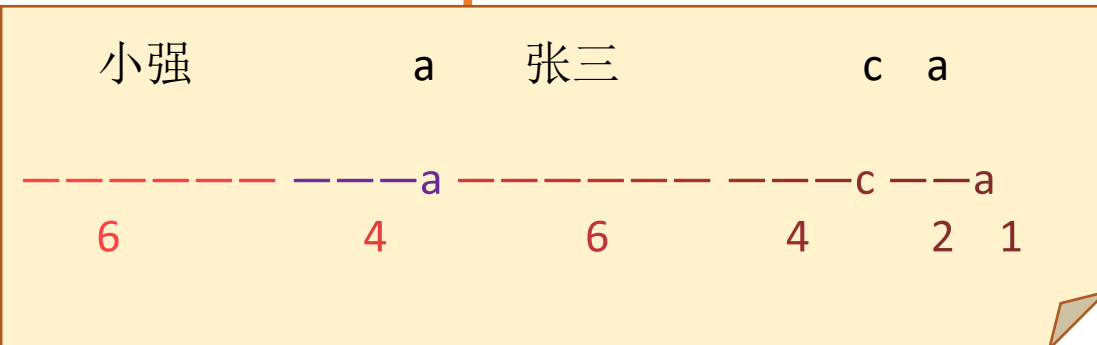
方法	说 明
<code>public void write(byte[] b) throws IOException</code>	将 len 个字节从指定 byte 数组写入到此文件，并从偏移量 off 处开始。
<code>public final void writeInt(int v) throws IOException</code>	四个字节将 int 写入该文件，先写高字节。写入从文件指针的当前位置开始
<code>public int read(byte[] b) throws IOException</code>	将最多 b.length 个数据字节从此文件读入 byte 数组
<code>public int read() throws IOException</code>	从此文件中读取一个数据字节。以整数形式返回此字节，范围在 0 到 255 (0x00-0x0ff)。
<code>public long getFilePointer() throws IOException</code>	返回此文件中的当前偏移量
<code>public void seek(long pos) throws IOException</code>	置到此文件开头测量到的文件指针偏移量，在该位置发生下一个读取或写入操作

# 知识点7：掌握随机访问文件流

## • RandomAccessFile向文件中写入数据

```
RandomAccessFile raf = new RandomAccessFile("e:/bb.txt", "rw");
raf.write("小强".getBytes()); // 6个字节
raf.writeInt(97); // 4个字节

raf.write("张三".getBytes()); // 6个字节
raf.writeInt(99); // 4个字节
raf.seek(22);
raf.writeByte(97);
raf.close();
```



```
RandomAccessFile raf = new RandomAccessFile("e:/bb.txt", "r");
byte[] b = new byte[6];
raf.read(b);
String name = new String(b);
```

```
int age = raf.readInt();
System.out.println("name=" + name + " age=" + age);
```

//得到指针的位置

```
long lo = raf.getFilePointer();
System.out.println("指针指向的位置="+lo);
raf.seek(22);
```

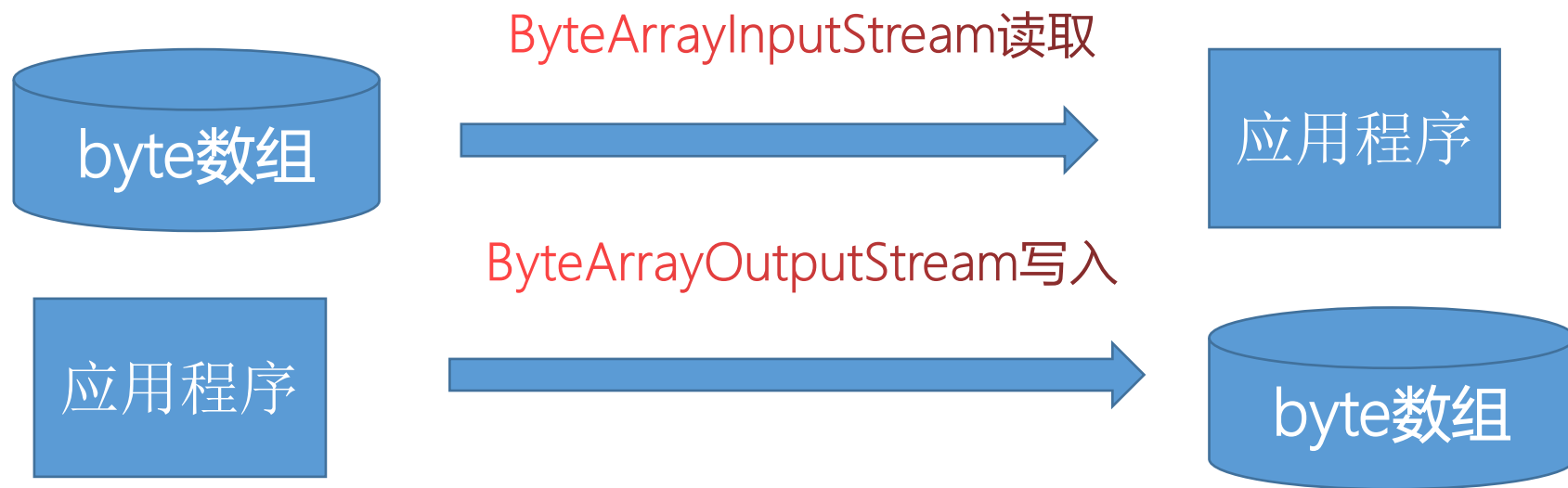
```
int hao = raf.read();
System.out.println(hao);
```



## 知识点8：掌握字节数组流



- 字节数组流：提供了针对于字符数组 `byte []` 的标准的IO操作方式
- 位于 `java.io` 包
- `ByteArrayInputStream` 和 `ByteArrayOutputStream`
  - `ByteArrayInputStream` 将会给一个 `byte buf[]` 提供标准的IO操作方式
  - `ByteArrayOutputStream` 则是将数据写入到内部的字节数组中



## 知识点8：掌握字节数组流



- ByteArrayInputStream 和 ByteArrayOutputStream
- ByteArrayOutputStream 写入的是自己内部的字节数组，属于内存数据,不涉及任何资源,所以不需要 close

构造方法	说 明
public ByteArrayInputStream(byte[] buf)	创建一个 ByteArrayInputStream，使用 buf 作为其缓冲区数组
public ByteArrayOutputStream()	创建一个新的 byte 数组输出流

ByteArrayOutputStream类方法	说 明
public byte[] toByteArray()	创建一个新分配的 byte 数组。其大小是此输出流的当前大小，并且缓冲区的有效内容已复制到该数组中

## 知识点8：掌握字节数组流



- 从文件中写出字节,
- 存到一个文件中

```
/*
 * 测试ByteArrayOutputStream
 * 将文件转化为字节数组
 */
public static byte[] TestBos(){
    byte[] flush=new byte[1024];
    int len=-1;
    try{
        BufferedInputStream bis=new BufferedInputStream(new FileInputStream("e:/bb.txt"));
        ByteArrayOutputStream baos= new ByteArrayOutputStream();
        while((len=bis.read(flush))!=-1){
            baos.write(flush, 0, len);
        }
        return baos.toByteArray();
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

```
/*
 * 测试ByteArrayInputStream 将字节数组输出到文件
 */
public static void TestBis(byte[] fileByteArray) {
    int len = -1;
    byte[] flush = new byte[1024];
    try {
        ByteArrayInputStream bais = new ByteArrayInputStream(fileByteArray);
        BufferedOutputStream bos = new BufferedOutputStream(new FileOutputStream("e:/cc.txt", true));
        while ((len = bais.read(flush)) != -1) {
            bos.write(flush, 0, len);
            bos.flush();
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

生成一个新的数组