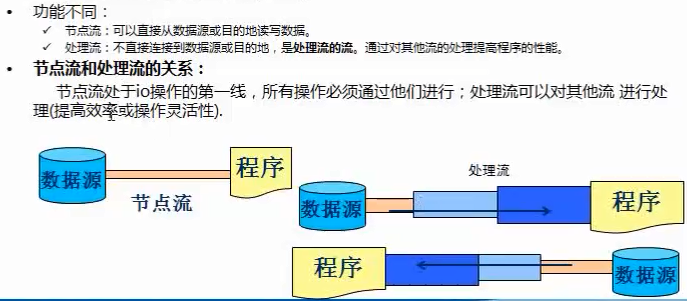
Java中的IO流技术

1. **IO流（Input & OUtput）主要是**用来处理**设备**之间的数据传输，对数据进行**读写操作**。Java对数据的操作都是通过**流**的方式，IO流主要分为**输入流**和**输出流**，**IO流**都是对象。Java中的**IO流对象**都存在于**java.io包**中。
2. **IO流的理解：**输入流，输出流，就像水流一样，需要建立**管道**，IO流对象就像管道一样，是一点一点的进行传输；在**数据源与目的地**之间建立了一个管道。
3. **I/O流操作的目的**：从**数据源**（多种多样，如**文件、键盘、网络**等等）当中**读取数据**，以及将数据**写入**到**数据目的地**（如**文件、屏幕、网络**等等）当中。
4. **常见的输入输出设备：**
5. **输入设备（数据源）：文件、键盘、网络等；**
6. **输出设备（数据目的地）：文件、屏幕（控制台）、网络等。**
7. **IO流的分类：**
8. **按照操作方向分类： 输入流和输出流；（方向都是以程序为中心）**
9. **按照操作数据类型分类： 字节流和字符流；**
10. **按照操作过程（或功能）分类： 节点流和处理流；**

**节点流：包裹源头，最原始的流。**

**处理流：增加新的功能，提高性能。 处理流一定要在节点流之上。**

**处理流：缓冲流和转换流等。**



1. **输入流和输出流：**输入和输出是基于**Java程序**而言的，而Java程序一般又是在**内存**中运行，所以也可以看做是基于内存而言的，IO系统的数据流向是以**Java程序**为参照物。将内存之外的数据**读入内存**称为读，即输入；将内存中的数据**写到内存之外**，称为写，即输出。输入的方法就是**read方法**，输出的方法就是**write方法**或者**print方法**。
2. **字节流和字符流：**
3. **字节流可以操作任何数据文件。所有的数据都是由字节组成的，都可以由字节流进行操作，然而由于存在不同的编码形式，同一个字符对应不同的字节，为了对字符操作方便，后来又推出了字符流，字符流只可以直接对字符进行操作，如常见的文本文件，不可以操作图像、音视频等文件。**
4. **字符流 = 字节流 + 编码表。**字节流在操作**字符对应的字节数据**时，**先查询指定的码表**，获取对应的**字符**，再对**这个字符**进行操作。简单地说，字符流就是**字节流+编码表**。字符流是以**字符**为处理单位的。
5. 如果要操作**文本数据**，建议优先考虑**字符流**，因为字符流就是基于这个目的出现的。
6. **码表：**由于不同的国家有不同的文字，同时又有自己的**文字编码表**，为了国际统一化，推出了国际统一编码 Unicode编码。中国的汉字编码表为**GBK**。
7. Java中所有的**IO流对象**都在**java.io包**中。主要是根据**字节流**和**字符流**分为两大派。字节流和字符流各有自己的**两个顶层类**。
8. 字节流的两个顶层类： **InputStream** 和 **OutputStream**；
9. 字符流的两个顶层类： **Reader** 和 **Writer**。

注意： 这四个顶层类都是直接继承Object类的，都是**抽象类（abstract）**，不可以new出对象，但可以利用**多态性**，其引用变量可以**指向所有子类的对象**。这四个抽象类都具有自己的构造方法，（抽象类的构造方法是用来为**子类初始化**的）。

1. **Java中IO流对象命名特点：** **子类名**都是以**父类名**作为后缀的。IO流名称的前缀代表此**IO流的功能**，其后缀代表其**操作的数据类型和操作方向**（字节OR 字符，输入OR输出）。
2. **字节流**介绍：
3. **InputStream** : **抽象类**，直接继承于Object。**重点是掌握3个read方法。**
   1. 只有一个空参数的**构造方法**：



* 1. 只有一个抽象方法，是空参数的**read（）方法**，返回int类型数据即读取的字节数据，无数据则返回**-1**。定义为抽象方法是让子类去**覆写（override）**。



* 1. 共定义了**3个read方法，返回值都是int类型，但意义不一样，无数据是都是返回 -1** ；**功能分别为: 读取一个字节、读取数组大小的字节、读取指定长度的字节并写入字节数据的指定位置**。
     1. **read（） 方法**：无参数，抽象方法，返回的是**读取的字节**，转换成了**int类型**；



* + 1. **read(byte[] b)方法：返回读取的字节个数；**



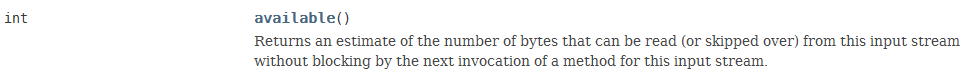
* + 1. **read（byte[] b，int off，int len）方法：返回读取的字节个数；**



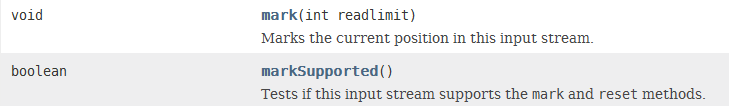
* 1. 只要是一个**IO流**，就会有**close（）方法**：

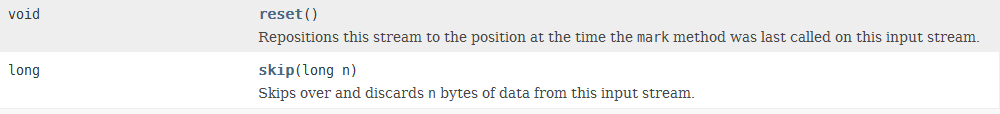


* 1. **available（）方法：字节输入流特有的方法，字符输入流没有；**



* 1. 其它方法：





1. **OutputStream：抽象类，直接继承于Object类；**
   1. 只有一个空参数的**构造方法**：



* 1. 只有一个抽象方法：**write(int b)方法**：只可以写入一个**字节数据**。返回值为void，就是需要子类覆写此方法的。



* 1. 共有3个write方法：**返回值都是void。功能分别为：写入一个字节、写入字节数组的全部字节、写入字节数据的一部分。**
     1. **write(int b)方法**:写入一个字节；返回void。



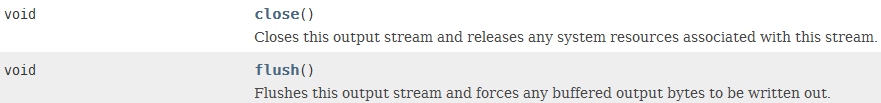
* + 1. **write(byte[] b)方法**: 把整个字节数组**全部写入**；返回void。



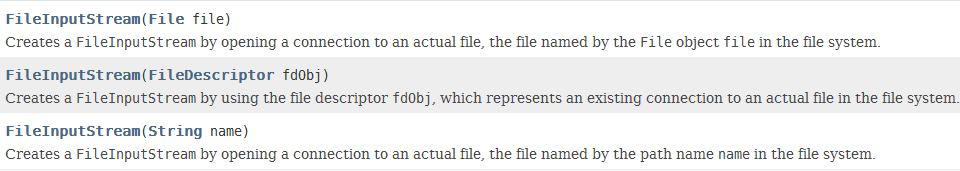
* + 1. **write(byte[] b , int off,int len)方法**: 把**字节数组的一部分**写入；返回void。



* 1. **flush() 方法 和close()方法**：

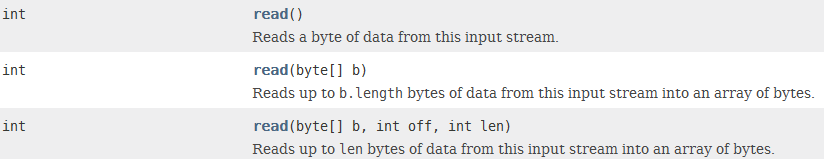


1. 所有的**字节输入流**都是**InputStream的子类**，所有的**字节输出流**都是**OutputStream的子类**。
2. 操作**文件数据**的两个子类是**FileInputStream** 和**FileOutputStream**；子类FileInputStream是用来读取**文件**的字节数据，子类FileOutputStream 是用来向**文件**写入字节数据。一般地，**文本文件**利用字符流FileReader和FileWriter，而音视频、图像等文件利用**字节流**。
3. **FileInputStream类**：
   * 1. **构造方法：有三个，File对象或者File路径及名称。**



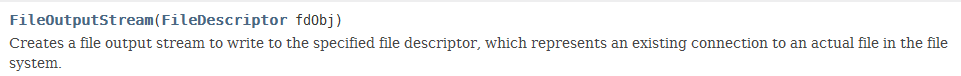
* + 1. **read方法：3个，与InputStream中的功能一样。**

**分别为读取一个字节，读取字节数组大小的字节到字节数组中，读取指定长度的字节到数组的特定位置。**



1. **FileOutputStream类**：
   1. **构造方法：5个。**







**与文件字节输入流相比较，文件字节输出流多了个功能，就是续写原来存在的文件，只需要在生成对象时，把append设置为true即可实现文件续写。**





* 1. **write方法：3个，返回void值，分别为 写入一个字节、把整个字节数组都写入、写入字节数组的一部分。**

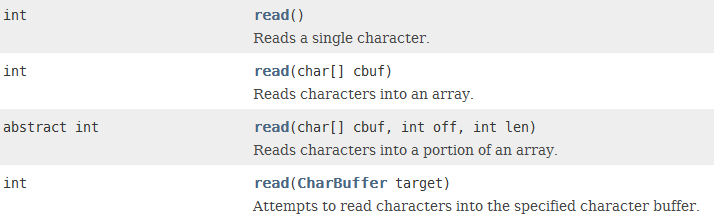






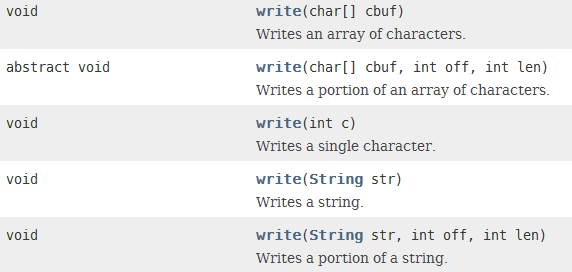
1. **字符流：Reader和Writer父类。**
2. **抽象类Reader下直接继承的子类主要是BufferedReader和InputStreamReader。注意FileReader是InputStreamReader转换流的直接子类。**
3. **抽象类Writer下直接继承的子类主要是BufferedWriter和OutputStreamWriter。注意FileWriter是OutputStreamWriter转换流的直接子类。**
4. **对于字符流就是字节流+编码表，其中最重要的方法read和write方法，字节流操作的是字节和字节数组，字符流就是字符和字符数组。**
5. **注意：由于字符有的占1个字节有的占2或3个字节，Java采用的是Unicode编码，所以每个字符都占据2个字节，即char类型占2个字节，int类型占4个字节，可以接收UTF-8的3个字节的字符，所以read方法都是用int类型进行接收，所有的read方法返回类型都是int。**
6. **字节流的write方法参数：byte、int、byte[]，字符流的write方法参数有：char[] 、int.**
7. **字符流:**
8. **Reader类的核心方法：read方法**

**与InputStream类的read方法差别在于一个是字符数组，一个是字节数组。**

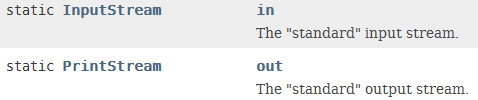


1. **Writer类的核心方法：write方法**

**与OutputStream类的write方法差别在于一个是字符数组，一个是字节数组。**



1. **System.in 这个输入流属于字节输入流，返回一个InputStream，只有一个，关闭后，再次创建就不能使用了，System.out 返回一个PrintStream，也是只有一个，关闭后，再次创建也不能使用。**



1. **装饰设计模式:** Java中io系统中使用最多的就是**装饰着模式**。“装饰者”模式中存在**装饰者和被装饰者**，装饰者是给被装饰者添加**新的功能**的。**被装饰者**作为参数传进装饰着的构造方法中，从而实现对其的**功能扩展**。
2. **BufferedReader 、BufferedWriter、BufferedInputStream、BufferedOutputStream。带有缓冲区的IO流，对流进行包装，实现高效性。**
3. **ButteredReader类**中的**readLine方法**：读取一行，返回字符串，当读到尾部时，返回**null**。**注意**：read方法读到结束时，返回的是 -1。

**BufferedWriter类中有个newLine方法，可以切换到一个新行。**



1. **针对BufferedWriter和BufferedOutputStream中的newLine方法就是把下面的一句话进行了封装：**

static final String **LINE\_SEPARATOR** = System.getProperty("line.separator");

**newline()方法**只是在带Buffered的输出流中有，然而上句话可以在任何情况下使用，因此若不使用带有缓冲区的输出流时，要学会使用上面的这句话。

1. **BufferedReader是直接继承于Reader，BufferedWriter是直接继承于Writer。InputStreamReader是直接继承于Reader，OutputStreamWriter是直接继承于Writer；但是FileReader是直接继承于InputStreamReader，不是直接继承于Reader的；FileWriter是直接继承于OutputStreamWriter，而不是直接继承于Writer的。注意:FileInputStream 是直接继承于InputStream的，而FileOutputStream是直接继承于OutputStream的，因为操作的就是字节，不需要转换。**
2. **待读取数据的文件必须存在，而要写入数据的文件可以不存在，会自动创建。**
3. **读文件操作**：如果文件不存在，则会出现**java.io.FileNotFoundException**异常。
4. **写文件操作**：如果文件不存在，就会**创建此文件**；如果文件存在，就会覆盖原来的数据（首先将原来数据清空，再写入新的数据）。如果需要**续写原文件**，可以在输出流的构造方法中**append设置为true**即可实现续写。
5. **（1）FileWriter是OutputStreamWriter转换流的子类，是对OutputStreamWriter的装饰，可以**直接操作**文本文件**。转换流OutputStreamWriter可以指定**任意的编码表**，然而如果**OutputStreamWritrer + 平台默认的编码表**，就是**FileWriter，如果没有指定编码表，直接使用FileWriter即可，但是如果指定编码表（非平台默认编码表），只能使用OutputStreamWriter了**。

**即 FileWriter fw = new FileWriter("e:\\11.txt");//已经封装好，直接就是将字节流转成字符流。**

**OutputStreamWriter osw = new**

**OutputStreamWriter(new FileOutputStream("e:\\11.txt"),"gbk");//将字节流转成字符流**

**上面两句代码的功能是等同的，平台默认编码表就是GBK。**

**（2）FileReader是InputStreamReader转换流的子类，转换流InputStreamReader可以指定任意的编码表，然而InputStreamReader + 平台默认的编码表就是FileReader。**

**FileReader fr = new FileReader("e:\\11.txt");**

**InputStreamReader isr = new InputStreamReader(new FileInputStream("e:\\11.txt"));**

**总结：因此FileWriter和FileReader就是操作文件的字节流+默认的编码表，也就是按照默认的编码表操作文件的便捷类而已。**

如果需要指定特定的编码表，**FileWriter和FileReader就不行了，必须使用转换流。**

1. **IO流的异常处理方法：**

IO流都需要抛出异常信息，而**try{}又是闭合的**，在try{}定义的引用变量在try外部无法使用，所以一般都是在**try{} catch{}之外**创建一个**空引用**，如**FileWriter fw = null；** 然后在try{} 括号内对引用进行**赋予对象**。一般在finally{} 语句中加入close方法，但是**close方法**需要单独再次try{} catch{} ，以防关闭此文件时**出现异常**。如：

**FileInputStream fis = null;//创建引用变量**

**try {**

**fis = new FileInputStream("e:\\ddd");**

**//codes.......**

**} catch (FileNotFoundException e) {**

**e.printStackTrace();**

**}**

**finally {**

**try {**

**fis.close();**

**} catch (IOException e) {**

**e.printStackTrace();**

**}**

**}**

1. **装饰设计模式：**对一组对象的功能进行进行增强时，就可以是使用该模式进行问题的解决。

**装饰和继承的区别：**

相同点：**都可以对功能进行扩展**；

不同点：装饰比继承更为灵活。**装饰类和被装饰类必须同属于同一个接口或者父类。 被装饰类以参数传进装饰类即可。**

如果对一个体系的任何类都需要进行功能的扩展，都要去创建一个子类，就会使体系变得十分的臃肿，这样显然不好。如果这些类需要拓展的功能都是一样的，利用继承，是使这功能与具体的类相结合，但是可以从另外一个角度想，将这种需要拓展的功能进行**单独的封装**，哪个类需要拓展这样的功能，就把这个类和此功能进行关联。

例如：

BufferedWriter 是装饰类，而Writer就是被装饰类。

BufferedReader 是装饰类，而Reader就是被装饰类。

1. **注意：windows中的回车符是两个字符\r\n，其中\r的ascii码为13，\n的ascii码为10. 另外制表符\t 的ascii码为9.**

System.***out***.println((**int**)'\r');//输出为13

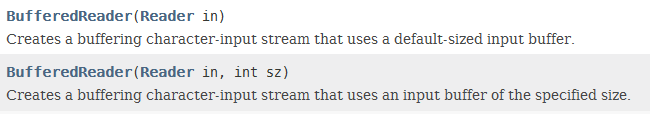
System.***out***.println((**int**)'\n'); //输出为10

System.***out***.println((**int**)'\t'); //输出为9

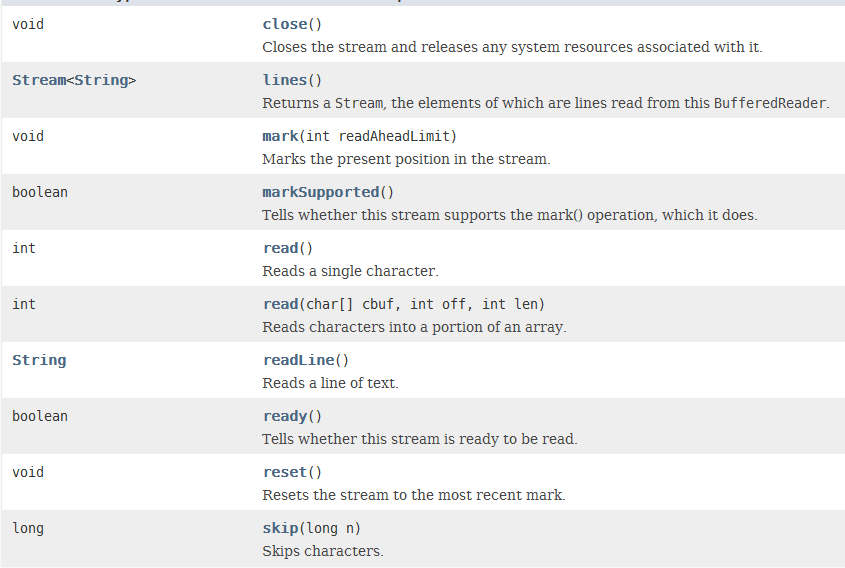
1. 字符流缓冲区：**BufferedReader 和 BufferedWriter**

**为了提高效率，引入了字符流的缓冲区。缓冲区一定要有被缓冲的对象，故构造方法都不是空参数的。**

1. **BufferedReader：是类Reader的子类；**
   1. **构造方法：**



* 1. **成员方法：**



* 1. **特有方法：readLine（）方法**

**FileReader fr = new FileReader("text2.txt");**

**BufferedReader br = new BufferedReader(fr);**

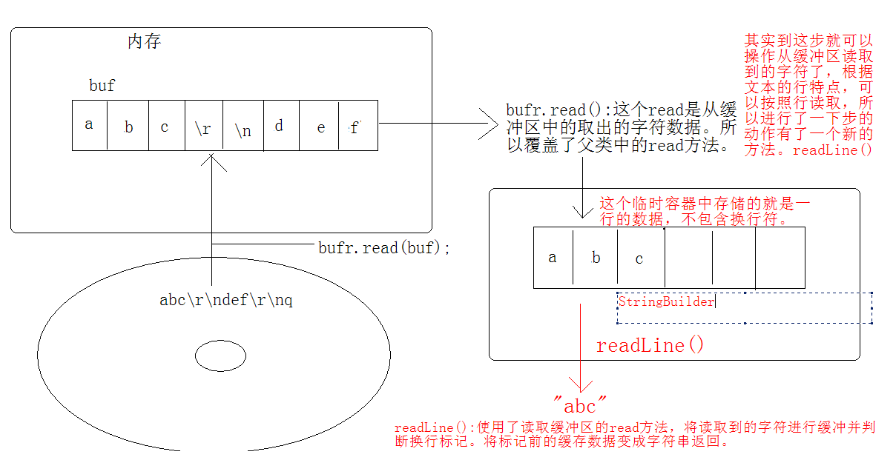
**String str = null;**

**while((str = br.readLine()) != null)**

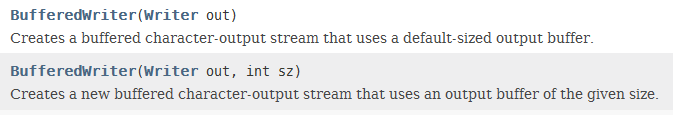
**System.out.println(str);**

**④** 缓冲区BufferedReader中的read方法与父类Reader类中read方法的区别：缓冲区BufferedReader中的read方法是**重写了**父类Reader类中read方法，前者是从**缓冲区**中读取数据，而后者是直接从**内存**中读取。到了这一步，就可以直接操作缓冲区中的数据了，例如实现一行一行的读取。

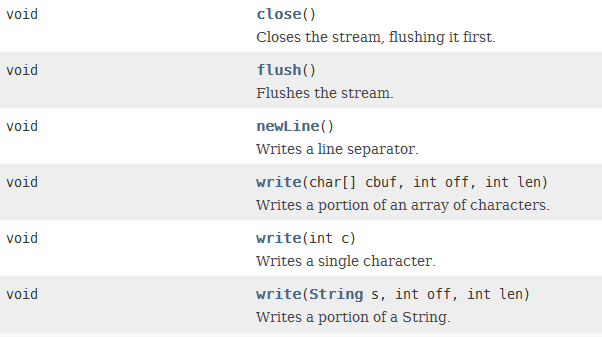
* 1. 对于**readLine方法**，就是使用了读缓冲区中的read方法，将读取到的字符进行缓冲，（可以利用StringBuffer或StringBuilder来实现），并判断是否到了换行标记，将换行符前的缓冲数据（不包括换行符）变成字符串返回。



1. **BufferedWriter：是类Writer的子类。**
   1. 构造方法：



* 1. 成员方法：

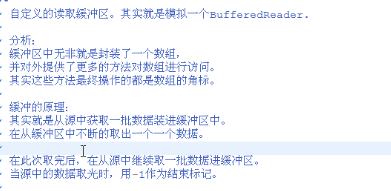


* 1. **特有方法：newLine（）方法**就是把下面的语句进行了封装。但是此方法只有在此对象中有。下面的语句在任何地方都可以使用。

public static final String LINE\_SEPARATOR = System.getProperty("line.separator");

System.out.print(LINE\_SEPARATOR);//词句与System.out.println();功能相同。

1. **自定义读取缓冲区：**



**package** sone.November29.am;

**import** java.io.FileReader;

**import** java.io.IOException;

**public** **class** MyBufferedReader {

**private** FileReader r;

**private** **int** pos;

**private** **int** count;

**private** **char**[] buf = **new** **char**[1024];

**public** MyBufferedReader(FileReader r) {

**super**();

**this**.r = r;

}

**public** **int** myRead() **throws** IOException {

**if**(count == 0) {

count = r.read(buf);

pos = 0;

}

**if**(count < 0)

**return** -1;

**char** ch = buf[pos++];

count--;

**return** ch;

}

**public** String myReadLine() **throws** IOException {

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

**int** ch = 0;

**while**((ch = myRead())!= -1) {

**if** (ch == '\t')

**continue**;

**if**(ch == '\n')

**return** sb.toString();

sb.append((**char**)ch);

}

**if**(sb.length() != 0)//防止最后一行读取不到，因为如果判断不到‘\n'就不会返回字符串

**return** sb.toString();

**return** **null**;

}

**public** **void** myClose() **throws** IOException {

//下午7:38:27

r.close();

}

}

1. **InputStreamReader 是FileReader的父类，是字节流转成字符流的桥梁，称为转换流。可以指定编码表，charset，解码。FileReader之所以能够操作字符，就是因为从InputStreamReader中继承了方法。同理，FileWriter之所以能够操作字符，也是因为从OutputStreamWriter中继承了方法。且 FileWriter和FileWriter 除了有自己的构造方法外，没有特有方法。**

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

InputStream is = System.in;

InputStreamReader isr = new InputStreamReader(is);

BufferedReader br = new BufferedReader(isr);

String line = null;

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

if("over".equals(line))

break;

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

}

}

一个中文占用两个字节，而一个中文算作是一个字符，故利用**InputStreamReader读取中文汉字的时候，会一次读取两个字节，因为一个中文对应两个字节。上面的例子，输入：山东科技大学 输出 ：山东科技大学**

InputStream is = System.***in***;

InputStreamReader isr = **new** InputStreamReader(is);

**int** ch = isr.read();

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

输入 你，则输出 你。

1. **OutputStreamWriter 是FileWriter的父类，也是字符流转成字节流的桥梁，称为转换流。可以指定编码表，charset，编码。**

/\* InputStream is = System.in;

InputStreamReader isr = new InputStreamReader(is);

BufferedReader br = new BufferedReader(isr);\*/

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

/\* OutputStream os = System.out;

OutputStreamWriter osw = new OutputStreamWriter(os);

BufferedWriter bw = new BufferedWriter(osw);\*/

**BufferedWriter bw =new BufferedWriter(new OutputStreamWriter(System.out));**

**String line = null;**

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

**if("over".equals(line)) {**

**break;**

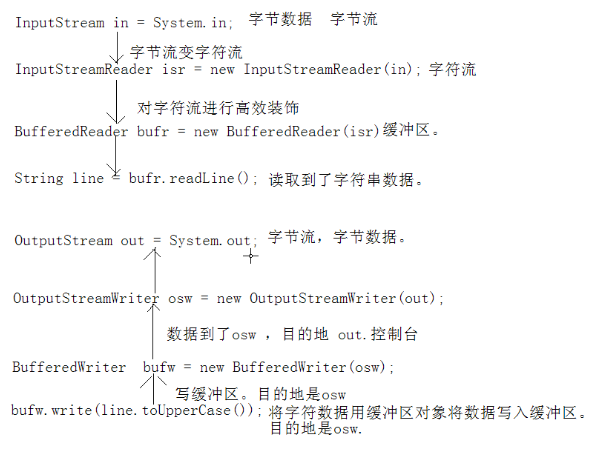
**}**

**bw.write(line.toUpperCase());**

**bw.newLine();**

**bw.flush();**

**}**解析：



1. **三句代码转换成一句代码：**

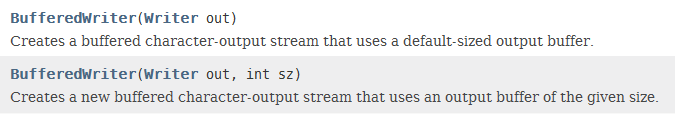
**键盘录入：**

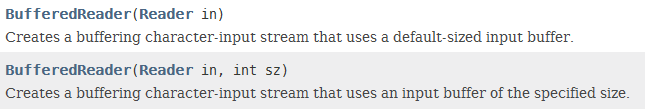
**BufferedReader bufr = new BufferedReader(new InputStreamReader(System.in));**

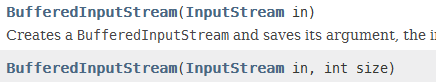
**BufferedWriter bufw= new BufferedWriter(new OutputStreamWriter(System.out));**

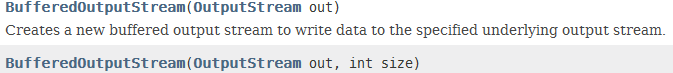
1. **节点流与处理流之间的关系：处理流是用来装饰节点流的，给节点流添加新的功能的。节点流就是被装饰者，而处理流就是装饰者。如BufferedReader 就可以用来装饰FileReader的，不仅可以装饰FileReader，而且可以装饰键盘读入，网络读入等等。**

**BufferedReader 、BufferedWriter、BufferedInputStream、BufferedOutputStream等，参数都是相应的被装饰的对象。**





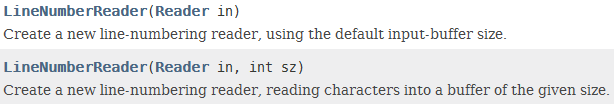




1. **FileOutputStream 的write方法，直接把数据写到目的地，不需要flush刷新方法，但是close方法是需要的。FileInputStream类： 中有个方法available（） ，可以返回此文件可读的字节数目，使用时要慎用，一般适用于小文件。**
2. **LineNumberReader 类**：是**BufferedReader**的子类，对Reader进行装饰。

**LineNumerInputStream已经过时了，不再使用，也就不用学习了。**

1. **构造方法有：**



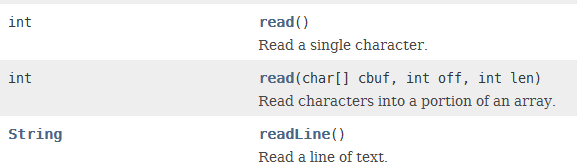
1. **成员方法：**







**read方法：**



**reset方法：**



1. **什么时候使用转换流（InputStreamReader和OutputStreamWriter）：**
2. **源或者目的对应的设备是字节流，但是操作的确是文本数据，可以使用转换流作为转换桥梁；**
3. **操作文本时，涉及到指定编码表时，要使用转换流。**
4. **流的操作规律：**

**（总结此规律的原因：流对象很多，开发时不知道选用那个流对象。）**

**在开发时能够很清楚的选用哪些流对象，需要掌握四个明确：**

1. **明确是源还是目的（汇）**
2. **源：只有两个体系InputStream 和 Reader；**
3. **目的（汇）：只有两个体系OutputStream 和 Writer；**
4. **明确数据是否是纯文本文件：**
   1. **源： 若是纯文本文件：Reader ；若不是，选用 InputStream**
   2. **目的（汇）：若是纯文本文件：Writer ；若不是，选用 OutputStream**

**前两个明确，就可以确定了需求中具体使用哪个体系。**

1. **明确具体的设备。**
   1. **源设备：**

**硬盘 ： File**

**键盘 ：System.in**

**内存 ： 数组**

**网络 ：Socket流**

* 1. **目的设备：**

**硬盘：File**

**控制台：System.out**

**内存：数组**

**网络：Socket流**

1. **是否需要额外功能。**

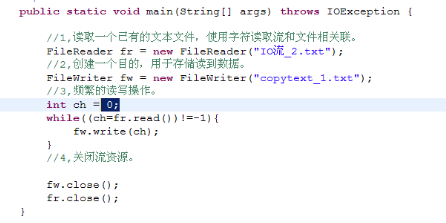
**是否需要高效（缓冲区）：是的话，就需要添加Buffered**

1. **练习题：**
2. 将**内存中的数据**写到**硬盘中指定的文件**里。
   * 1. 硬盘数据的基本体现就是**文件**，所以找到一个处理文件的类即可。
     2. 创建一个可以向文件中写入字符数据的**字符输出流对象**；既然写数据，就必须在创建对象时就指定某个文件（用于存储数据的目的地），也就是说这个**输出流类不能有空参数的构造方法**。
     3. 利用Writer对象的**write方法**，写入数据。将数据写入到临时存储缓冲区中。利用**flush方法**进行刷新，可以将数据写到文件中。close方法关闭流和关闭资源执行时，先进行刷新，将缓冲区中的数据写入到文件中。flush和close就相当于保存命令和关闭命令，关闭命令执行之前会提醒是否保存。
3. 将**c盘**中的一个文本文件复制到**d盘**中。

复制原理：读取数据，粘贴：写数据。 **连读带写。**

**思路：（1） 读取数据源；（2）将读取的源数据写到指定位置。**

1. **读取一个写一个：**



1. **用数组作为缓冲区:**

**FileReader fr = null;**

**FileWriter fw = null;**

**try {**

**fr = new FileReader("text1.txt");**

**fw = new FileWriter("copy2.txt");**

**// 创建一个容器，用于缓存读取到的数据**

**char[] buf = new char[BUFFER\_SIZE];**

**// 定义一个变量，存取读取到的数据的字节数**

**int len = 0;**

**for (; (len = fr.read(buf)) != -1;) {**

**fw.write(buf, 0, len);**

**}**

**} catch (IOException e) {**

**throw new RuntimeException("读写失败");**

**}**

**finally {**

**if (fr != null)**

**try {**

**fr.close();**

**} catch (IOException e) {**

**e.printStackTrace();}}**

**if (fw != null)**

**try {**

**fw.close();**

**} catch (IOException e) {**

**e.printStackTrace();}**

1. **利用Buffered：**

**BufferedReader bufr = new BufferedReader(new FileReader("e:\\from.txt"));**

**BufferedWriter bufw = new BufferedWriter(new FileWriter("e:\\to2.txt"));**

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

**int len ;**

**while((len = bufr.read(buf,0,buf.length)) != -1) {**

**bufw.write(buf,0,len);**

**bufw.flush();**

**}**

**bufw.close();**

**bufr.close();**

**或者：**

**BufferedReader bufr = new BufferedReader(new FileReader("e:\\from.txt"));**

**BufferedWriter bufw = new BufferedWriter(new FileWriter("e:\\to2.txt"));**

**String line = null;**

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

**bufw.write(line);**

**bufw.flush();**

**bufw.newLine();**

**}**

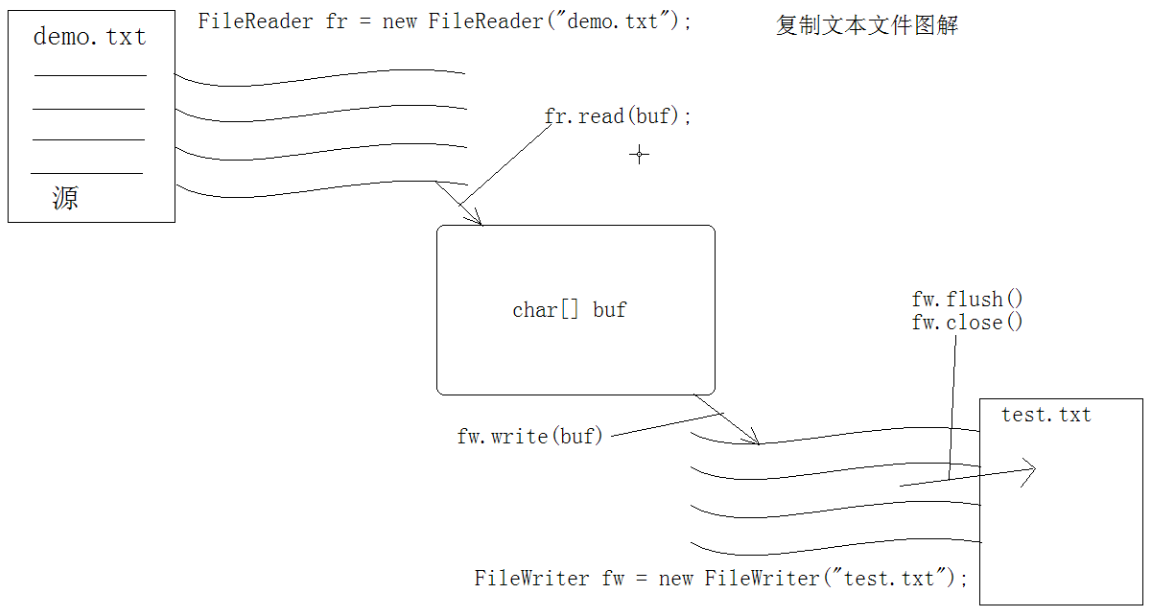
**bufw.close();**

**bufr.close();**

1. **复制文本文件图解:**

**flush 方法可以把数据从缓冲区送入文件中。**

**close方法可以完成flush方法的功能。**



1. **大文件的读写：**

实例一 字节流：//类的导入

import java.io.\*;

class Test6{

public static void main(String[] args){

//声明输入输出流的引用

FileInputStream fil = null;

FileOutputStream fos =null;

FileOutputStream fos2 = null;

try{

//生成代表输入输出流的对象

fil = new FileInputStream("g:/javalianxi/io/from.txt");

fos = new FileOutputStream("g:/javalianxi/io/to.txt");

fos2 = new FileOutputStream("g:/javalianxi/io/to2.txt");

//生成一个字节数组

byte[] buffer = new byte[100];

while(true){

int temp=fil.read(buffer,0,buffer.length);

if(temp == -1){

break;

}

fos.write(buffer,0,temp);

fos2.write(buffer,0,temp);

}

// fos.write(buffer,0,temp);

// String s = new String(buffer); //把字节型数组转换成字符串

// s = s.trim();

// System.out.println(s);

/\* for (int i =0;i<buffer.length;i++){

System.out.println(buffer[i]);

}\*/

}

catch( Exception e){

System.out.println(e);

}

finally{

try{

fil.close();

fos.close();

}

catch(Exception e){

System.out.println(e);

}

}

}

}

实例二 **字符流**：

import java.io.\*;

class Test7{

public static void main(String[] args){

FileReader fr = null;

FileWriter fw = null;

try{

fr = new FileReader("g:/javalianxi/io/from.txt");

fw = new FileWriter("g:/javalianxi/io/to2.txt");

char[] buffer = new char[1024];

while(true){

int temp = fr.read(buffer,0,buffer.length);

if (temp == -1){

break;

}

fw.write(buffer,0,temp);

}

}

catch(Exception e){

System.out.println(e);

}

finally{

try{

fr.close();

fw.close();

}

catch(Exception e){

System.out.println(e);

}

}

}

}

1. **完成以下需求：**
2. **将键盘录入的数据写入到一个文件中。**

**把System.out 换成new FileOutputStream("e:\\copy2.txt") 就可以了。即：**

**BufferedWriter bw = new BufferedWriter(new OutputStreamWriter(new FileOutputStream("e:\\copy2.txt")));**

1. **将一个文本文件内容显示在控制台上。**

**把System.in 换成new FileInputStream("e:\\copy2.txt") 就可以了。即：**

**BufferedReader br = new BufferedReader(new InputStreamReader(new FileInputStream("e:\\copy2.txt")));**

1. **将一个文件中的内容复制到另外一个文件中。**

**综合（1）（2）即可：**

**BufferedReader br = new BufferedReader(new InputStreamReader(new FileInputStream("e:\\copy2.txt")));**

**BufferedWriter bw = new BufferedWriter(new OutputStreamWriter(new FileOutputStream("e:\\copy4.txt")));**

1. **练习题1. 复制一个文本文件：**

分析过程

(1) 读取： Reader InputStream 写入： Writer OutputStream

(2) 纯文本：则 读 Reader 写 Writer

(3) 硬盘 ，源是文件，目的 硬盘 文件。

FileReader fr = new FileReader("aaa.txt");

FileWriter fw = new FileWriter("bbb.txt");

(4)需要高速： 缓冲区**并**

BufferedWriter bufw = new BufferedWriter(fw);

BufferedReader bufr = new BufferedReader(fr);

1. **练习题2. 读取键盘录入信息，写到一个文件中。**

**/\* 47． 练习题2. 读取键盘录入信息，并写到一个文件中。**

**\* 1.分析过程**

**(1) 读取： Reader InputStream 写入： Writer OutputStream**

**(2) 纯文本：则 读 Reader 写 Writer**

**(3) 键盘 ，源是 System.in 目的 硬盘 文件。**

**InputStream in = System.in;**

**这样做很麻烦，将读取的字节数据转换成字节串，再由字符流操作。**

**(4)需要： 转换**

**将字节流转换成字符流，因为已经明确源是Reader，这样操作文本数据会很便捷。所以将已有的字节流转成字符流，字节 ——》字符，InputStreamReader InputStreamReader isr = new InputStreamReader(System.in);**

**FileWriter fw = new FileWriter("bbb.txt");**

**需要高效：**

**BufferedReader bufr = new BufferedReader(new InputStreamReader(System.in));**

**BufferedWriter bufw = new BufferedWriter(new FileWriter("bbb.txt"));**

**\*/**

1. **练习题3.将一个文本文件数据显示在控制台上**

/\* \* 1.分析过程

(1) 读取： Reader InputStream 写入： Writer OutputStream

(2) 纯文本：则 读 Reader 写 Writer

(3) 源 硬盘 文件 目的 显示器 （控制台） System.out。

FileReader fr = new FileReader("aaa.txt");

OutputStream os = System.out;

(4)需要： 转换

将字符流转换成字节流，因为已经明确源是Writer

字符——》字节， OutputStreamWriter

FileReader fr = new FileReader("aaa.txt");

OutputStreamWriter osw = new OutputStreamWriter(System.out);

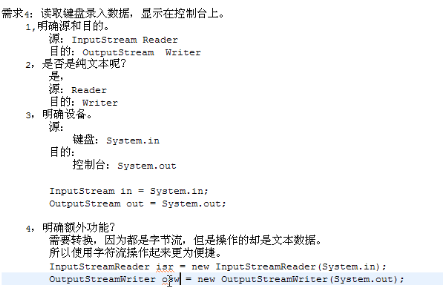
需要高效：

BufferedReader bufr = new BufferedReader(new FileReader("aaa.txt"));

BufferedWriter bufw = new BufferedWriter(new OutputStreamWriter(System.out));

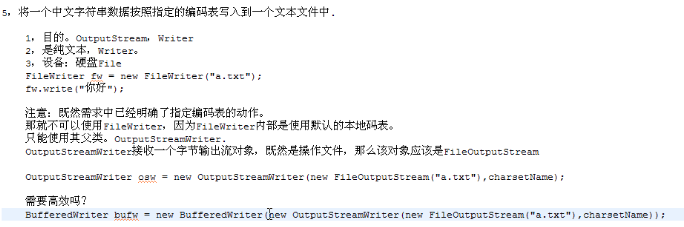
\*/

1. **练习题4. 读取键盘录入数据，显示在控制台上。**

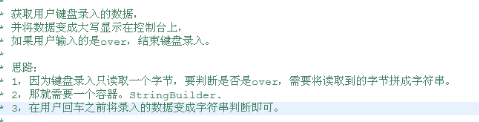




1. **练习题5. 将一个中文字符串数据按照指定的字符串编码表写入到一个文件中。（转换流的最大好处：就是可以指定不同的编码表）**



1. **读取键盘录入：**



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

**//下午10:03:43**

**//1.创建容器**

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

**//2.获取键盘读取流**

**InputStream in = System.*in*;**

**//3.记录读取到的字节，并循环获取**

**int ch = 0;**

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

**//4. 在存储好之前需要判断是否是换行标记，因为换行标记不存储，**

**if(ch == '\r')**

**continue ;**

**if(ch == '\n') {**

**String temp = sb.toString();**

**if("over".equals(temp))**

**break;**

**System.*out*.println(temp.toUpperCase());**

**sb.delete(0,sb.length());//清空**

**}**

**else**

**sb.append((char)ch);**

**}**

**}**