## 字符流

Reader

|--InputStreamReader

|--FileReader：专门用于处理文件的字符读取流对象

Writer

|--OutputStreamWriter

|--FileWriter：专门用于处理文件的字符写入流对象

#### Reader API

1. int read()

读取一个字符。返回的是读到的那个字符。如果读到流的末尾，返回-1。

1. int read(char[])

将读到的字符存入指定的数组中，返回的是读到的字符个数，也就是往数组里

装的元素的个数。如果读到流的末尾，返回-1。

1. close()

读取字符其实用的是window系统的功能，就希望使用完毕后，进行资源的释

放。

#### Writer API

1. write(char)

将一个字符写入到流中。

1. write(char[])

将一个字符数组写入到流中。

1. write(String)

将一个字符串写入到流中。

1. flush()

刷新流，将流中的数据刷新到目的地中，流还存在。

1. close()

关闭资源。在关闭前会先调用flush()，刷新流中的数据去目的地。然后流

关闭。

#### FileReader API

1. FileReader(String fileName)

在读取流对象初始化的时候，必须要指定一个被读取的文件。如果该文件不存在会发生 FileNotFoundException。

#### FileWriter API

1. FileWriter(String fileName)

在指定位置，创建一个文件。如果该文件已存在，将会被覆盖。

1. FileWriter(String fileName, boolean append)

当传入的 boolean 类型值为 true 时，会在指定文件末尾处进行数据的续写。

#### 注意事项

当指定绝对路径时，定义目录分隔符有两种方式：

1. 反斜线。但是一定要写两个\\，new FileWriter("c:\\demo.txt");
2. 斜线/写一个即可。new FileWriter("c:/demo.txt");

### 字符流的缓冲区

在使用缓冲区对象时，要明确，缓冲的存在是为了增强流的功能而存在，所以在建立缓冲区对象时，要先有流对象存在。

#### BufferedReader API

1. BufferedReader(Reader reader)
2. newLine ()

跨平台的换行符。

#### BufferedWriter API

1. BufferedWriter(Writer writer)
2. String readLine()

一次读一行，到行标记时，将行标记之前的字符数据作为字符串返回。当读到末尾时，返回null。

### 转换流

InputStreamReader API

1. InputStreamReader(InputStream)

通过该构造函数初始化，使用的是本系统默认的编码表GBK。

1. InputStreamReader(InputStream, String charSet)

通过该构造函数初始化，可以指定编码表。

OutputStreamWriter API

1. OutputStreamWriter(OutputStream, String charSet)

通过该构造函数初始化，可以指定编码表。

## 字节流

InputStream

|--FileInputStream

|--BufferedInputStream

OutputStream

|--FileOutputStream

|--BufferedOutputStream

InputStream API

1. int read()
2. read(byte[])
3. close()
4. available()

获取关联的文件的字节数。

OutputStream API

1. writer(String)
2. close()

FileInputStream API

1. FileInputStream(String fileName)

FileOutputStream API

1. FileOutputStream(String fileName)

清单文件2-1

BufferedInputStream bufferedInputStream = new BufferedInputStream(new FileInputStream(“demo.png”));

BufferedOutputStream bufferedOutputStream = new BufferedOutputStream(new FileOutputStream(“demo\_copy.png”));

// int size = bufferedInputStream.available();

// byte[] buf = new byte[size];

// bufferedInputStream.read(buf);

// bufferedOutputStream.writer(buf.toString());

int size = 0;

while((size = bufferedInputStream.read() != -1) {

bufferedOutputStream.write(size);

}

bufferedOutputStream.close();

bufferedInputStream.close();