# 字符流

## 概述

字符流是可以直接读写字符的IO流

字符流读取字符, 就要先读取到字节数据, 然后转为字符. 如果要写出字符, 需要把字符转为字节再写出.

## FileReader

### 新增构造

FileReader(File file)

在给定从中读取数据的 File 的情况下创建一个新 FileReader。

FileReader(FileDescriptor fd)

在给定从中读取数据的 FileDescriptor 的情况下创建一个新 FileReader。

FileReader(String fileName)

在给定从中读取数据的文件名的情况下创建一个新 FileReader。

文件字节输入流

FileReader **char**[] b = **new** **char**[20];

## FileWriter

### (4) 文件字节输入流

FileWriter

# 文件拷贝

## 直接

**public** **static** **void** demo1() **throws** FileNotFoundException, IOException {

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

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

**int** c;

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

fw.write(c);

}

fr.close();

fw.close(); //Writer类中有一个2k的小缓冲区,如果不关流,就会将内容写到缓冲区里,关流会将缓冲区内容刷新,再关闭

}

## 定义小数组

**public** **static** **void** demo3() **throws** FileNotFoundException, IOException {

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

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

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

**int** len;

**while**((len = fr.read(arr)) != -1) { //将文件上的数据读取到字符数组中

fw.write(arr,0,len); //将字符数组中的数据写到文件上

}

fr.close();

fw.close();

}

## 缓冲区

**public** **static** **void** demo4() **throws** FileNotFoundException, IOException {

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

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

**int** c;

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

bw.write(c);

}

br.close();

bw.close();

}

# 注意

什么情况下使用字符流？

字符流也可以拷贝文本文件, 但不推荐使用. 因为读取时会把字节转为字符, 写出时还要把字符转回字节.

\* 程序需要读取一段文本, 或者需要写出一段文本的时候可以使用字符流

\* 读取的时候是按照字符的大小读取的,不会出现半个中文

\* 写出的时候可以直接将字符串写出,不用转换为字节数组

字符流是否可以拷贝非纯文本的文件？

不可以拷贝非纯文本的文件

\* 因为在读的时候会将字节转换为字符,在转换过程中,可能找不到对应的字符,就会用?代替,写出的时候会将字符转换成字节写出去

\* 如果是?,直接写出,这样写出之后的文件就乱了,看不了了

# 缓冲流

## 缓冲思想

字节流一次读写一个数组的速度明显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(装饰设计模式后面讲解)，所以提供了字节缓冲区流

把节点流包入缓冲流

## BufferedReader和

\* BufferedReader的read()方法读取字符时会一次读取若干字符到缓冲区, 然后逐个返回给程序, 降低读取文件的次数, 提高效率

读写文件的方法

readLine() BufferedReader流读取文本行

## BufferedWriter类

\* BufferedWriter的write()方法写出字符时会先写到缓冲区, 缓冲区写满时才会写到文件, 降低写文件的次数, 提高效率

write(String s,int off,int len) BufferedWriter流把字符串s写到文件中

newLine(); BufferedWriter流向文件写入一个回行符

//具体实现文本复制

String str = **null**;

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

bw.write(str);

bw.newLine(); //换行

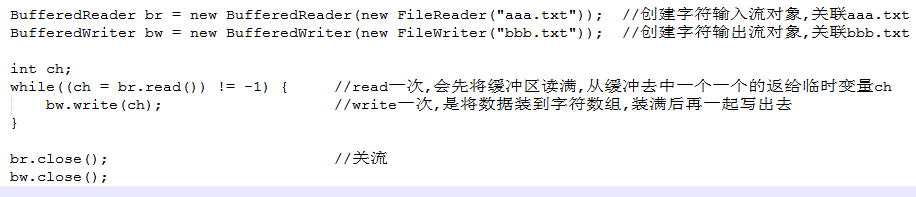
bw.flush();

}

newLine()与\r\n的区别

newLine()是跨平台的方法

\r\n只支持的是windows系统



# LineNumberReader

LineNumberReader是BufferedReader的子类, 具有相同的功能, 并且可以统计行号

\* 调用getLineNumber()方法可以获取当前行号

\* 调用setLineNumber()方法可以设置当前行号

**public** **class** Demo5\_LineNumberReader {

/\*\*

\* **@param** args

\* **@throws** IOException

\*/

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

LineNumberReader lnr = **new** LineNumberReader(**new** FileReader("zzz.txt"));

String line;

lnr.setLineNumber(100);

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

System.*out*.println(lnr.getLineNumber() + ":" + line);

}

lnr.close();

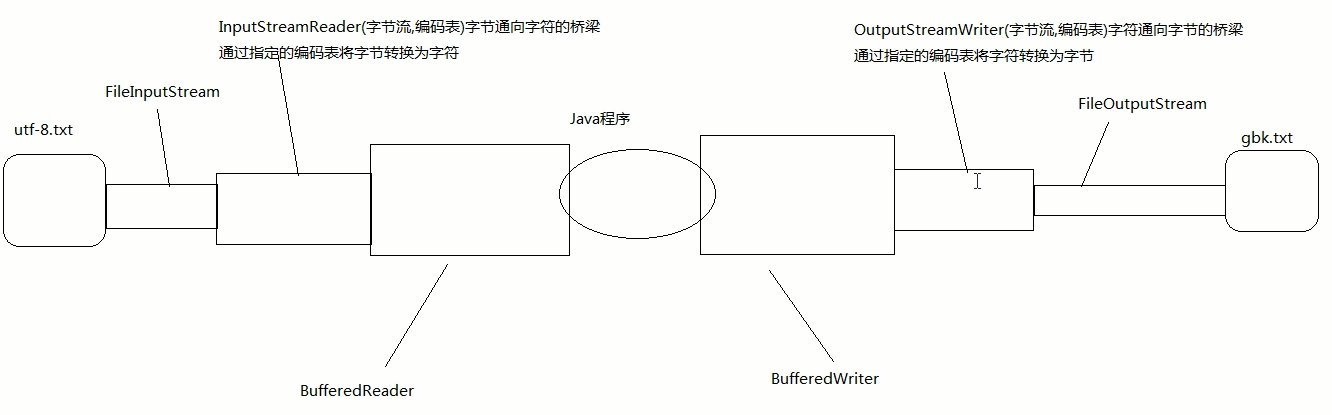
}

}

# 使用指定码表读写字符

FileReader是使用默认码表读取文件, 如果需要使用指定码表读取, 那么可以使用InputStreamReader(字节流,编码表)

\* FileWriter是使用默认码表写出文件, 如果需要使用指定码表写出, 那么可以使用OutputStreamWriter(字节流,编码表)



## 5.转换流 ——字节流🡪字符流 处理字符编码

### (1) 字符编码

解码：字节数组🡪字符串 InputStreamReader

编码：字符串🡪字节数组 OutputStreamWriter

### (2)一般处理

InputStreamReader isr = **new** InputStreamReader(fis,"UTF-8");

OutputStreamWriter osw = **new** OutputStreamWriter(fos,"UTF-8");

**public** **static** **void** demo2() **throws** UnsupportedEncodingException,

FileNotFoundException, IOException {

InputStreamReader isr = **new** InputStreamReader(**new** FileInputStream("utf-8.txt"), "uTf-8"); //指定码表读字符

OutputStreamWriter osw = **new** OutputStreamWriter(**new** FileOutputStream("gbk.txt"), "gbk"); //指定码表写字符

**int** c;

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

osw.write(c);

}

isr.close();

osw.close();

}

**public** **static** **void** demo3() **throws** UnsupportedEncodingException,

FileNotFoundException, IOException {

BufferedReader br = //更高效的读

**new** BufferedReader(**new** InputStreamReader(**new** FileInputStream("utf-8.txt"), "utf-8"));

BufferedWriter bw = //更高效的写

**new** BufferedWriter(**new** OutputStreamWriter(**new** FileOutputStream("gbk.txt"), "gbk"));

**int** c;

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

bw.write(c);

}

br.close();

bw.close();

}

# 案例

## 将文本反转

**public** **class** Test1 {

/\*\*

\* **@param** args

\* 将一个文本文档上的文本反转,第一行和倒数第一行交换,第二行和倒数第二行交换

\*

\* 分析:

\* 1,创建输入输出流对象

\* 2,创建集合对象

\* 3,将读到的数据存储在集合中

\* 4,倒着遍历集合将数据写到文件上

\* 5,关流

\* **@throws** IOException

\*

\* 注意事项:

\* 流对象尽量晚开早关

\*/

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

//改写后是尽量晚开早关

// 1,创建输入输出流对象

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

//2,创建集合对象

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

//3,将读到的数据存储在集合中

String line;

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

list.add(line);

}

br.close(); //关流

//4,倒着遍历集合将数据写到文件上

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

**for**(**int** i = list.size() - 1; i >= 0; i--) {

bw.write(list.get(i));

bw.newLine();

}

//5,关流

bw.close();

}

}

## 获取一个文本上每个字符出现的次数,将结果写在times.txt上

**public** **class** Test2 {

/\*\*

\* **@param** args

\* 获取一个文本上每个字符出现的次数,将结果写在times.txt上

\*

\* 1,创建带缓冲区的输入流对象

\* 2,创建双列集合对象,目的是把字符当作键,把字符出现的次数当作值

\* 3,通过读取不断向集合中存储,存储的时候要判断,如果不包含这个键就将键和值为1存储,如果包含就将键和值加1存储

\* 4,关闭输入流

\* 5,创建输出流对象

\* 6,将结果写出

\* 7,关闭输出流

\* **@throws** IOException

\*/

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

//1,创建带缓冲区的输入流对象

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

//2,创建双列集合对象,目的是把字符当作键,把字符出现的次数当作值

HashMap<Character, Integer> hm = **new** HashMap<>();

//3,通过读取不断向集合中存储,存储的时候要判断,如果不包含这个键就将键和值为1存储,如果包含就将键和值加1存储

**int** c;

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

**char** ch = (**char**)c;

/\*if(!hm.containsKey(ch)) {

hm.put(ch, 1);

}else {

hm.put(ch, hm.get(ch) + 1);

}\*/

hm.put(ch, !hm.containsKey(ch)? 1 : hm.get(ch) + 1);

}

//4,关闭输入流

br.close();

//5,创建输出流对象

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

//6,将结果写出

**for** (Character key : hm.keySet()) {

bw.write(key + "=" + hm.get(key));

}

bw.close();

}

}