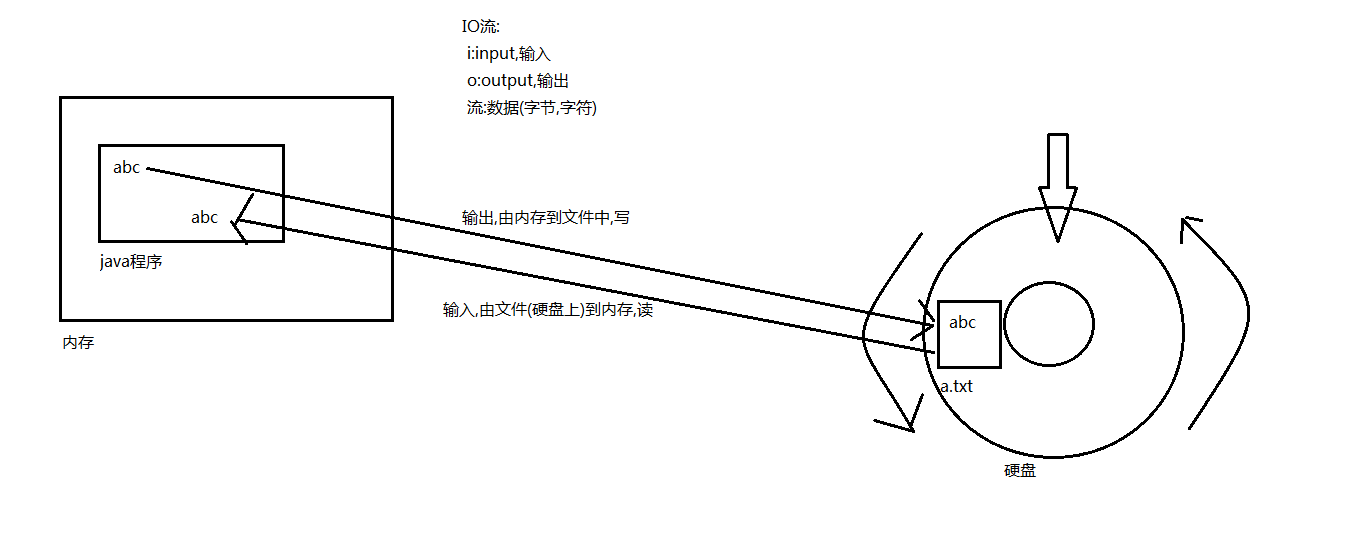
今日学习目标

* 能够说出IO流的分类和功能
* 能够使用FileWriter写数据到文件
* 能够明确FileWriter写数据使用两种路径的区别
* 能够说出FileWriter中关闭和刷新方法的区别
* 能够使用FileWriter写数据的5个方法重载
* 能够使用FileWriter写数据实现换行和追加写
* 能够使用FileReader读数据
* 能够使用FileReader和FileWriter完成一次读写一个字符复制文本文件
* 能够使用FileReader读数据一次一个字符数组
* 能够使用FileReader和FileWriter完成一次读写一个字符数组复制文本文件
* 能够明确字符缓冲流的作用和基本用法
* 能够使用缓冲流复制文本文件
* 能够使用缓冲流的特殊功能
* 能够使用缓冲流特殊功能复制文本文件

# FileWriter和FileReader

## IO流概述和分类

* IO流用来处理设备之间的数据传输
  + 文件复制(U盘拷贝)
  + 上传文件(微信图片,QQ空间图片)
  + 下载文件(迅雷)
* IO流分类
  + 字符输出流
    - FileWriter
  + 字符输入流
    - FileReader
* 图解



## FileWirter

### FileWriter写数据

* 构造方法
  + FileWriter(String fileName) 传递一个文件名称
* 成员方法
  + void write(String str) 往文件中写入一个字符串。
  + void flush() 刷新该流的缓冲，把内存缓冲区中的数据刷新到文件中。
  + void close() 关闭此流，但要先刷新它。
* 输出流写数据的步骤：
  + 1.创建输出流对象
  + 2.调用输出流对象的写数据的方法
  + 3.释放资源

**public** **class** Demo01FileWriter {

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

//创建输出流对象

FileWriter fw = **new** FileWriter("d:\\a.txt");

/\*

\* 创建输出流对象做了哪些事情:

\* A:调用系统资源创建了一个文件

\* B:创建输出流对象

\* C:把输出流对象指向文件

\*/

//调用输出流对象的写数据的方法

//写一个字符串数据

fw.write("IO流你好");

//数据没有直接写到文件，其实是写到了内存缓冲区

fw.flush();

//释放资源

//通知系统释放和该文件相关的资源

fw.close();

//while(true) {}

}

}

### FileWriter写数据路径问题及关闭和刷新方法的区别

* 路径:
  + 相对路径：相对当前项目而言的，在项目的根目录下(a.txt)
  + 绝对路径：以盘符开始的路径(d:\\a.txt)
* close()和flush()方法的区别：
  + flush():刷新缓冲区。流对象还可以继续使用。
  + close():先刷新缓冲区，然后通知系统释放资源。流对象不可以再被使用了。

**public** **class** Demo02FileWriter {

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

//创建输出流对象

//FileWriter fw = new FileWriter("d:\\a.txt");

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

//调用输出流对象的写数据方法，并刷新缓冲区

fw.write("helloworld");

fw.flush();

fw.write("java");

fw.flush();

//释放资源

fw.close();

//Stream closed

//fw.write("javaee");

//fw.flush();

}

}

### FileWriter写数据的5个方法

* void write(String str):写一个字符串数据
* void write(String str,int index,int len):写一个字符串中的一部分数据, index:开始索引,len:写几个
* void write(int ch):写一个字符数据,这里写int类型的好处是既可以写char类型的数据，也可以写char对应的int类型的值。'a',97
* void write(char[] chs):写一个字符数组数据
* void write(char[] chs,int index,int len):写一个字符数组的一部分数据, index:开始索引,len:写几个

**public** **class** Demo03FileWriter {

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

//创建输出流对象

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

//void write(String str):写一个字符串数据

fw.write("abcde");

//void write(String str,int index,int len):写一个字符串中的一部分数据, index:开始索引,len:写几个

fw.write("abcde",0,5);

fw.write("abcde",1,3);

//void write(int ch):写一个字符数据,这里写int类型的好处是既可以写char类型的数据，也可以写char对应的int类型的值。'a',97

fw.write('a');

fw.write(97);

//void write(char[] chs):写一个字符数组数据

**char**[] chs = {'a','b','c','d','e'};

//fw.write(chs);

//void write(char[] chs,int index,int len):写一个字符数组的一部分数据, index:开始索引,len:写几个

fw.write(chs,0,5);

fw.write(chs,2,3);

//释放资源

fw.close();

}

}

### FileWriter写数据之换行和追加写

* 续写问题
  + FileWriter(String fileName, boolean append)
  + 构造放中参数的作用:
    - 第一个参数:写入文件的目的地
    - 第二个参数:append作用

true:可以续写

false:不能续写,覆盖之前的文件

* 2.换行问题
  + windows:\r\n
  + linux:\n
  + mac:\r
  + 换行符可以写在第一个数据的结尾,也可以写在第二个数据的开头

## FileReader

### FileReader读数据

* 构造方法
  + FileReader(String fileName) 传递要读取的文件名称
* 成员方法
  + int read() 读取单个字符并返回
  + int read(char[] cbuf) 一次读取一个字符数组的数据，返回的是实际读取的字符个数
* 输入流读文件的步骤：
  + 1.创建输入流对象
  + 2.调用输入流对象的读数据方法
  + 3.释放资源

**public** **class** Demo01FileReader {

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

//创建输入流对象

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

FileReader fr = **new** FileReader("Demo01FileWriter.java");

//调用输入流对象的读数据方法

//int read():一次读取一个字符

**int** ch;

/\*

\* while循环的条件表达式一共做了3件事情

\* 1:fr.read() 读取一个字符

\* 2:ch=fr.read() 把读取的字符赋值给ch

\* 3:ch != -1 判断ch是否为-1

\*/

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

//System.out.println(ch);

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

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

}

//释放资源

fr.close();

}

}

* 注意:如果构造方法中的文件不存在,会抛异常
  + java.io.FileNotFoundException: fr.txt (系统找不到指定的文件。)

### 复制文本文件: 一次读写一个字符复制文本文件

* 操作步骤:
  + 1.创建FileReader对象,并且绑定数据源
  + 2.创建FileWriter对象,并且绑定数据目的地
  + 3.调用FileReader中读取字符的方法read
  + 4.调用FileWriter中写入字符的方法write
  + 5.释放资源

**public** **class** Demo01CopyFile {

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

//创建输入流对象

FileReader fr = **new** FileReader("Demo01FileWriter.java");

//创建输出流对象

FileWriter fw = **new** FileWriter("Copy.java");

//读写数据

**int** ch;

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

fw.write(ch);

}

//释放资源

fw.close();

fr.close();

}

}

### FileReader读数据一次一个字符数组

* int read(char[] cbuf) 一次读取一个字符数组的数据，返回的是实际读取的字符个数

**public** **class** Demo02FileReader {

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

//创建输入流对象

//FileReader fr = new FileReader("fr2.txt");

FileReader fr = **new** FileReader("FileWriterDemo.java");

//调用输入流对象的读数据方法

//int read(char[] cbuf)：一次读取一个字符数组的数据，返回的是实际读取的字符个数

**char**[] chs = **new** **char**[1024]; //这里可以是1024及其整数倍

**int** len;

/\*

\* while循环的条件表达式一共做了3件事情

\* 1:fr.read(chs) 把数据读取到数组中

\* 2:len=fr.read(chs) 把读取的有效个数赋值给len

\* 3:len != -1 判断读取的有效个数是否为-1

\*/

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

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

}

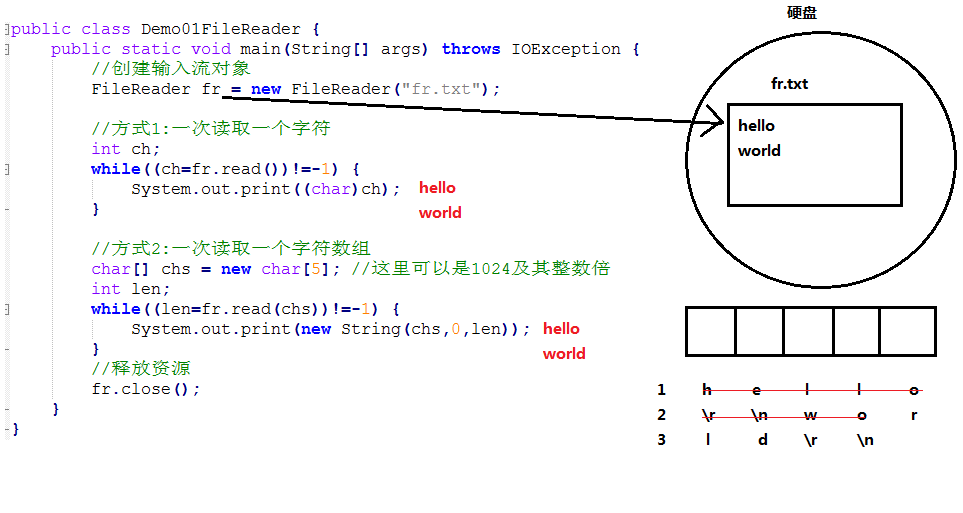
//释放资源

fr.close();

}

}

### -FileReader读取数据的两种方式图解



### 复制文本文件: 一次读写一个字符数组复制文本文件

**public** **class** Demo02CopyFile {

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

//创建输入流对象

FileReader fr = **new** FileReader("Demo01FileWriter.java");

//创建输出流对象

FileWriter fw = **new** FileWriter("Copy.java");

//读写数据

**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();

}

}

# BufferedWriter和BufferedReader

## 字符缓冲流

### 基本用法

* BufferedWriter:将文本写入字符输出流，缓冲各个字符，从而提供单个字符、数组和字符串的高效写入。
* BufferedReader:从字符输入流中读取文本，缓冲各个字符，从而实现字符、数组和行的高效读取。

**public** **class** Demo01BufferedStream {

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

//创建输出缓冲流对象

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

bw.write("hello");

//bw.flush();

bw.close();

//创建输入缓冲流对象

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

/\*

//一次读写一个字符

int ch;

while((ch=br.read())!=-1){

System.out.print((char)ch);

}

\*/

//一次读写一个字符数组

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

**int** len;

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

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

}

//释放资源

br.close();

}

}

### 缓冲流复制文本文件的两种方式

**public** **class** Demo02CopyFile {

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

//创建输入缓冲流对象

BufferedReader br =

**new** BufferedReader(**new** FileReader("Demo01FileWriter.java"));

//创建输出缓冲流对象

BufferedWriter bw = **new** BufferedWriter(**new** FileWriter("Copy.java"));

//读写数据

/\*

//一次读写一个字符

int ch;

while((ch=br.read())!=-1) {

bw.write(ch);

}

\*/

//一次读写一个字符数组

**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();

}

}

### 缓冲流的特殊功能

* BufferedWriter
  + void newLine():写一个换行符，这个换行符由系统决定
* BufferedReader
  + String readLine():一次读取一行数据，但是不读取换行符

**public** **class** Demo03BufferedStream {

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

BufferedWriter bw = **new** BufferedWriter(**new** FileWriter("bw2.txt"));

**for**(**int** x=0; x<10; x++) {

bw.write("hello"+x);

//bw.write("\r\n");

bw.newLine();

bw.flush();

}

bw.close();

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

String line;

/\*

\*while循环的条件表达式一共做了3件事情

\* 1:br.readLine() 读取一行

\* 2:line=br.readLine() 把读取的一行字符串赋值给line

\* 3:line != null 判断line是否为null

\*/

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

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

}

br.close();

}

}

### 缓冲流特殊功能复制文本文件

**public** **class** Demo04CopyFile {

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

//创建输入缓冲流对象

BufferedReader br =

**new** BufferedReader(**new** FileReader("Demo01FileWriter.java"));

//创建输出缓冲流对象

BufferedWriter bw = **new** BufferedWriter(**new** FileWriter("Copy.java"));

//读写数据

String line;

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

bw.write(line);

bw.newLine();

bw.flush();

}

//释放资源

bw.close();

br.close();

}

}

### IO流练习

* 把ArrayList集合中的字符串数据存储到文本文件
  + 每一个字符串元素作为文件中的一行数据

**public** **class** Demo02ArrayListToFileTest {

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

//创建集合对象

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

//往集合中添加字符串元素

list.add("hello");

list.add("world");

list.add("java");

//创建输出缓冲流对象

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

//遍历集合，得到每一个字符串元素，然后把该字符串元素作为数据写到文本文件

**for**(**int** i=0; i<list.size(); i++) {

String s = list.get(i);

bw.write(s);

bw.newLine();

bw.flush();

}

//释放资源

bw.close();

}

}

* 从文本文件中读取数据到ArrayList集合中，并遍历集合
  + 每一行数据作为一个字符串元素

**public** **class** Demo03FileToArrayListTest {

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

//创建输入缓冲流对象

BufferedReader br =

**new** BufferedReader(**new** FileReader("array.txt"));

//创建集合对象

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

//读取数据，每次读取一行数据，把该行数据作为一个元素存储到集合中

String line;

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

list.add(line);

}

//释放资源

br.close();

//遍历集合

**for**(**int** i=0; i<list.size(); i++) {

String s = list.get(i);

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

}

}

}