# 输入字符流 Reader

## 1 字节流 与 字符流 的区别

* + 字节流：字节流不会把读到的二进制数据转换成你能看的懂的数据，而是以ASCII码表的形式反馈给你。字节流每次读取1个字节(byte)的数据。
  + 字符流：字符流会把读取到的二进制的数据进行相应的编码与解码工作。 字符流 = 字节流 + 编码（解码）字符流每次读取1个字符(char)的数据。

注意：字符流每次只能读取1个字节，而每个中文占2个字节，因此使用字节流读取中文汉字时，会出现乱码问题。

### 1.1 编码表

ASCII：a 97 美国标准信息交换码。用一个字节的7位可以表示。

ISO8859-1： 拉丁码表。欧洲码表，用一个字节的8位表示。又称Latin-1(拉丁编码)或“西欧语言”。ASCII码是包含的仅仅是英文字母，并且没有完全占满256个编码位置，所以它以ASCII为基础，在空置的0xA0-0xFF的范围内，加入192个字母及符号，藉以供使用变音符号的拉丁字母语言使用。从而支持德文，法文等。因而它依然是一个单字节编码，只是比ASCII更全面。

GB2312： 英文占一个字节，中文占两个字节.中国的中文编码表。

GBK： 中国的中文编码表升级，融合了更多的中文文字符号。

Unicode： 国际标准码规范，融合了多种文字。所有文字都用两个字节来表示，Java语言使用的就是unicode。

UTF-8： 最多用三个字节来表示一个字符。

（我们以后接触最多的是iso8859-1、gbk、utf-8）

**ISO8859-1：**一个字节

**GBK：** 两个字节包含了英文字符和扩展的中文 ISO8859-1+中文字符

**UTF-8：** 万国码表，推行的。是1~3个字节不等长。英文存的是1个字节，中文存的是3个字节，是为了节省空间。

## 2 输入字符流

- - - - | Reader 输入字符流的基类，抽象类

- - - - - - - - | FileReader 读取文件的输入字符流

### 2.1 FileReader的用法

1.找到目标文件

2.搭建通道

3.读取数据

4.释放资源

### 2.2 案例

|  |
| --- |
| **public class** Dome1 {  **public static void** main(String[] args) {  *testReader*();// 测试缓冲字节流FileReader的方法  }   // 使用缓冲字符流读取文件  **public static void** testReader() {  // 1.找到目标文件  File file = **new** File(**"E:\\aa\\bb\\a.txt"**);   // 2.搭建通道  FileReader fileReader = **null**;  **try** {  fileReader = **new** FileReader(file);   // 3.读取数据  **char**[] chars = **new char**[1024];  **int** len = 0;  **while** ((len = fileReader.read(chars)) != -1) {  System.***out***.println(**new** String(chars, 0, len));  }  } **catch** (FileNotFoundException e) {  **throw new** RuntimeException(e);  } **catch** (IOException e) {  **throw new** RuntimeException(e);  } **finally** {  **try** {  // 4.释放资源  fileReader.close();  } **catch** (IOException e) {  **throw new** RuntimeException(e);  }  }  } } |

## 3 使用字符流读取文件数据，控制台出现乱码？

因为现在编译器(idea\eclipse等)被我们修改成了默认使用 utf-8编码格式，而windows系统的默认编码格式是gbk。因为编码格式的不同，从而导致了控制台显示FileReader读取文件数据在控制台显示乱码的现象。

**控制台出现乱码时，首先查看文件编码格式与控制台编码格式是否相同。**

# 2. 输出字符流 Writer

## 1. 输出字符流

- - - - | Writer 输出字符流的基类，抽象类

- - - - - - - - | FileWriter 向文件写出数据的输出字符流

## 2. FileWriter 输出字符流的使用步骤

1. 找到目标文件

2. 建立数据传输通道

3. 写出数据

4. 关闭资源

## 3. FileWriter 要注意的事项：

**1. （大坑）** FileWriter的基类 Writer维护了一个1024字节的数组，FileWriter.write(str)只是将字符串str写出到了这个数组中，要想将数据真正写出到硬盘，还需要调用flush或close方法，或者是这个数组被填满。

2. 如果目标文件不存在，新建一个文件，然后将数据写入到文件中。

3. 如果目标文件存在，先将原文件中的数据清空，然后才将需要的数据写入到文件中（即新数据覆盖掉旧数据）。

4. 如果要在目标文件原有数据的基础上追加数据，需要使用 new FileWriter(File file, boolean append)构造函数，第二个参数 append为true。

## 4. 案例

|  |
| --- |
| **public class** Dome1 {  **public static void** main(String[] args) {  // 1.找到目标文件  File file = **new** File(**"E:\\aa\\bb\\a.txt"**);   // 2.建立数据传输通道  FileWriter fileWriter = **null**;  **try** {  fileWriter = **new** FileWriter(file);   // 3.写出数据  /\*  FileWriter 字符输出流的一个大坑。  FileWriter的基类 Writer维护了一个1024字节的数组，FileWriter.write(str)只是将字符串str写出到了这个数组中，要想将数据真正写出到硬盘，还需要调用flush或close方法，或者是这个数组被填满。  \*/  String str = **"好好学习，天天想汉时关"**;  fileWriter.write(str);// 字符流具备解码的功能  } **catch** (IOException e) {  **throw new** RuntimeException(e);  } **finally** {  **try** {  // 4.关闭流资源  fileWriter.close();  } **catch** (IOException e) {  **throw new** RuntimeException(e);  }  }  } } |

## 5.字符流中，使用的默认编码格式是什么？

答：默认使用的是项目的编码格式。比如idea中 settings – editor – File Encoding – Project Encoding的编码格式是什么，io流的默认编码就是什么。

FileWriter 的getEncoding()方法可以获取编码名称，但是没有设置编码的方法，如果要设置编码格式，需要使用它的父类 OutputStreamWriter的构造方法 new OutputStream(File file, Charset charset)设置编码格式。

# 缓冲输入字符流BufferedReader

## 1．输入字符流:

- - - - | Reader 所有输入字符流的基类。 抽象类

- - - - - - - - | FileReader 读取文件字符串的输入字符流。

- - - - - - - - | BufferedReader 缓冲输入字符流。缓冲输入字符流出现的目的是为了提高读取文件的效率和拓展了FileReader的功能。 其实该类内部也是维护了一个字符数组

记住：缓冲流都不具备读写文件的能力。

## BufferedReader 的使用案例

**readLine()：一次读取一行数据，不能读\r\n**

|  |
| --- |
| **package** buffer;  **import** java.io.\*;  **public class** Dome1 {  // 程序入口main方法  **public static void** main(String[] args) {  File file = **new** File(**"E:\\aa\\bb\\Demo1.java"**);  // 使用字节写的readLine()方法读取文件数据  *testMyReadLine*(file);  System.***out***.println();  // 使用BufferedReader读取文件数据  *testBufferReader1*(file);   }   //测试BufferedReader类  **public static void** testBufferReader1(File file) {  // 2.搭建通道  FileReader fileReader = **null**;  BufferedReader bufferedReader = **null**;  **try** {  fileReader = **new** FileReader(file);//搭建硬盘与内存之间传输数据的通道  bufferedReader = **new** BufferedReader(fileReader);//缓冲类不具备读写能力   // 3.读取数据  String context = **null**;// 用来存储读取到的数据  **while** ((context = bufferedReader.readLine()) != **null**) {// readLine每次读一行，但是不读\r和\n  System.***out***.println(context);  }  } **catch** (FileNotFoundException e) {  **throw new** RuntimeException(e);  } **catch** (IOException e) {  **throw new** RuntimeException(e);  } **finally** {  **try** {  bufferedReader.close();  } **catch** (IOException e) {  **throw new** RuntimeException(e);  }  }  }   //测试自己写的readLine()方法  **public static void** testMyReadLine(File file) {  FileReader fileReader = **null**;  **try** {  fileReader = **new** FileReader(file);  String s = **null**;//存储读取到的字符串  **while** ((s = *myReadLine*(fileReader)) != **null**) {  System.***out***.println(s);  }  } **catch** (FileNotFoundException e) {  e.printStackTrace();  } **finally** {  **try** {  // 关闭资源  fileReader.close();  } **catch** (IOException e) {  **throw new** RuntimeException(e);  }  }  }   // 自己写一个readLine()方法  **public static** String myReadLine(FileReader fileReader) {  StringBuilder stringBuilder = **new** StringBuilder();//存储一行数据  **int** len = 0;//存储每次读取的数据  **try** {  // 循环对读取的字节进行判断，如果是普通字节，则追加到stringBuilder中，如果是\r 则重新读取，如果是\n则结束读取，并返回之前读取的所有字节组成的字符串  **while** (((len = fileReader.read()) != -1)) {  **if** (len == **'\r'**) {//如果读到的是\r，重新读取下一个字节  **continue**;  } **else if** (len == **'\n'**) {//如果读到的是\n，结束循环  **break**;  } **else** {  stringBuilder.append((**char**) len);//如果读取到普通字节，放入字符串中  }  }  **if** (len == -1) {  **return null**;//如果读取结果是-1，表示没有读取到任何数据，返回null值  }  } **catch** (IOException e) {  **throw new** RuntimeException(e);  } //资源交给调用者去关闭，否则会发生java.io.IOException: Stream closed  **return** stringBuilder.toString();//返回读取到的\n之前的字符串（也就是一行字符串）  } } |

# 缓冲输出字符流BufferedWriter

## 1. 输出字符流

- - - - | Writer所有输出字符流的基类，抽象类。

- - - - - - - - | FileWriter 向文件输出字符数据的输出字符流。

- - - - - - - - | BufferedWriter 缓冲输出字符流。缓冲输出字符流作用： 提高FileWriter的写数据效率与拓展FileWriter的功能。

BufferedWriter内部只不过是提供了一个8192长度的字符数组作为缓冲区而已，拓展了FileWriter的功能。

BufferedWriter提供了一个可以指定数组长度的构造方法。

Writer内部维护了一个1024字节长度的数组作为缓冲区，FileWriter继承了Writer类，所以FileWriter相当于内部也维护了一个1024数组的缓冲区。

## BufferedWriter案例

|  |
| --- |
| **public class** Dome2 {  **public static void** main(String[] args) {  // 1.指定目标文件  File file = **new** File(**"E:\\aa\\bb\\BufferedWriter.txt"**);  // 2.搭建通道  FileWriter fileWriter = **null**;  BufferedWriter bufferedWriter = **null**;  **try** {  fileWriter = **new** FileWriter(file);  bufferedWriter = **new** BufferedWriter(fileWriter);   // 3.传输数据  String content = **"大家好！！"**;//向硬盘文件输出的数据  bufferedWriter.write(content);// 向硬盘文件输出数据  bufferedWriter.newLine();// newLine()的作用就是向文件输出一个 \r\n，即输出一个换行符。  } **catch** (IOException e) {  **throw new** RuntimeException(e);  } **finally** {  // 关闭资源  **try** {  bufferedWriter.close();  } **catch** (IOException e) {  **throw new** RuntimeException(e);  }  }  } } |

# 使用缓冲字符模拟软件登录

|  |
| --- |
| **package** buffer;  **import** java.io.\*; **import** java.util.Scanner;  **public class** SimulationLogin {  **public static void** main(String[] args) {  File file = **new** File(**"E:\\aa\\bb\\user.txt"**);   **while** (**true**) {  System.***out***.println(**"欢迎使用模拟登录系统..."**);  System.***out***.println(**"请选择：A 注册 B 登陆 C 退出"**);  Scanner scanner = **new** Scanner(System.***in***);  String option = scanner.next();  **if** (**"a"**.equalsIgnoreCase(option)) {  // 注册  *reg*();  } **else if** (**"b"**.equalsIgnoreCase(option)) {  // 登录  *login*();  } **else if** (**"c"**.equalsIgnoreCase(option)) {  // 退出  System.***out***.println(**"成功退出模拟登陆系统..."**);  **return**;  } **else** {  System.***out***.println(**"输入有误，请重新输入..."**);  }  }  }    //注册功能  **public static void** reg() {  Scanner scanner = **new** Scanner(System.***in***);  System.***out***.print(**"请输入用户名:"**);  String username = scanner.next();// 用户名  String password = **null**;//密码  String user = **null**;//用户信息   File file = **new** File(**"E:\\aa\\bb\\user.txt"**);//找到目标文件  BufferedReader bufferedReader = **null**;  BufferedWriter bufferedWriter = **null**;   **try** {  // 搭建数据传输通道  bufferedReader = **new** BufferedReader(**new** FileReader(file));//读取数据对象  bufferedWriter = **new** BufferedWriter(**new** FileWriter(file, **true**));//输出数据对象  **boolean** isEsixt = **false**;//判断用户名是否存在的标识   String content = **null**;//用于存储读取的数据  // 判断用户名是否已存在  **while** ((content = bufferedReader.readLine()) != **null**) {//读取一行数据  String[] s = content.split(**" "**);//根据空格将读取的数据进行切分  **if** (username.equals(s[0])) {//判断读取的用户名与注册用户名是否相同  isEsixt = **true**;  **break**;  }  }  **if** (isEsixt) {// 存在提示重新注册  System.***out***.println(**"注册失败,用户名已存在,请重新输入新的用户名..."**);  isEsixt = **false**;  } **else** {// 不存在，提示用户注册成功，并将数据写入到硬盘文件中   System.***out***.print(**"请输入密码:"**);  password = scanner.next();// 密码  user = username + **" "** + password;//用户名与密码   bufferedWriter.write(user);//将数据写入硬盘文件  bufferedWriter.newLine();//换行  System.***out***.println(**"用户注册成功..."**);  System.***out***.println(**"用户信息：[username = "** + username + **", password = "** + password + **"]"**);  }  } **catch** (IOException e) {  **throw new** RuntimeException(e);  } **finally** {  **try** {  // 释放资源  bufferedWriter.close();  bufferedReader.close();  } **catch** (IOException e) {  **throw new** RuntimeException(e);  }  }  }    //登录功能  **public static void** login() {  Scanner scanner = **new** Scanner(System.***in***);  System.***out***.print(**"请输入用户名："**);  String username = scanner.next();// 用户名  System.***out***.print(**"请输入密码："**);  String password = scanner.next();// 密码  String userInfo = username + **" "** + password;// 用户信息   File file = **new** File(**"E:\\aa\\bb\\user.txt"**);//找到目标文件  BufferedReader bufferedReader = **null**;   **boolean** isExist = **false**;//用于标记用户是否存在的标识，默认不存在   **try** {  // 搭建数据通道  bufferedReader = **new** BufferedReader(**new** FileReader(file));  String user = **null**;// 用户存储从文件中读取的数据  // 读取文件数据，查询用户是否存在  **while** ((user = bufferedReader.readLine()) != **null**) {  **if** (userInfo.equals(user)) {  isExist = **true**;  **break**;  }  }   //判断用户是否存在  **if** (isExist) {  String[] s = user.split(**" "**);  System.***out***.println(**"欢迎用户 "** + s[0] + **" 成功登录模拟登录系统..."**);  } **else** {  System.***out***.println(**"用户名或密码不正确，请确认后重新输入..."**);  }  } **catch** (FileNotFoundException e) {  **throw new** RuntimeException(e);  } **catch** (IOException e) {  **throw new** RuntimeException(e);  } **finally** {  **try** {  // 释放资源  bufferedReader.close();  } **catch** (IOException e) {  **throw new** RuntimeException(e);  }  }  } } |