java基础知识 第10天

**【学习目标】理解、了解、应用、记忆**

通过今天的学习，参训学员能够：（解释的时候说出二级目标的掌握程度）

1. **【应用】IO流概述及FileWriter类使用**
2. 【了解】能够阐述IO流的作用
3. 【应用】能够独立完成利用FileWriter向文本文件中写数据以及追加数据
4. 【应用】能够理解并用代码测试write()的重载方法
5. **【应用】FileReader类使用和缓冲流使用**
6. 【应用】能够独立使用FileReader读数据一次读取一个字符
7. 【应用】能够写出利用FileReader和FileWrtie完成文本文件复制的两种方式
8. 【应用】能够使用缓冲流BufferedReader和BufferedWriter完成文本文件复制
9. 【应用】能够使用readLine()和newLine()完成文本文件复制
10. **【应用】IO流相关案例**
11. 【应用】能够熟练写出复制文本文件的五种方式
12. 【应用】能够独立完成将集合中的数据写到文本文件中
13. 【应用】能够将文本文件的数据读取到集合中

# IO流概述及FileWriter类使用

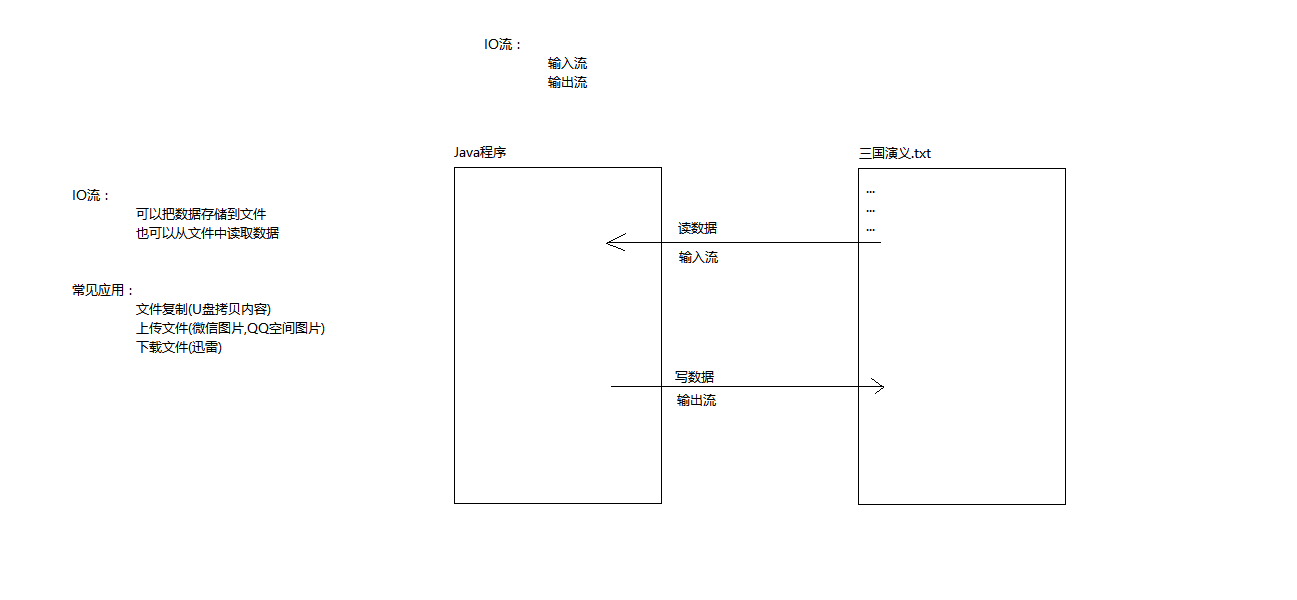
## IO流概述及分类

IO流用来处理设备之间的数据传输

Java对数据的操作是通过流的方式

Java用于操作流的类都在IO包中

流按流向分为两种：输入流，输出流



## FileWriter类使用

### FileWriter向文件中写数据

A:FileWriter向文件中写数据操作步骤:

a:使用FileWriter流关联文件

b:利用FileWriter的写方法写数据

c:利用FileWriter的刷新方法将数据从内存刷到硬盘上

d:利用FileWriter的关闭流方法将释放占用的系统底层资源

B:FileWriter方法:

构造方法

FileWriter(String fileName) 传入一个文件的路径

成员方法

void write(String str) 向文件中写str

void flush() 将内存中的数据刷新到文件中

void close() 关流释放系统底层资源

#### 案例代码一

|  |
| --- |
| **package** com.igeek\_02;  **import** java.io.FileWriter;  **import** java.io.IOException;  /\*\*  \* **@ClassName**: FileWriterDemo  \* **@Description**: FileWriter往文件中写数据  \* **@date** 2017年10月14日 下午1:25:20  \* Company www.igeekhome.com  \*  \* 需求：往文件中写数据  \* 写数据--输出流--FileWriter  \*  \* FileWriter:  \* FileWriter(String fileName):传递一个文件名称  \*  \* 输出流写数据的步骤：  \* A:创建输出流对象  \* B:调用输出流对象的写数据的方法  \* C:释放资源  \*/  **public** **class** FileWriterDemo {  **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注意事项

#### 案例代码二

|  |
| --- |
| **package** com.igeek\_02;  **import** java.io.FileWriter;  **import** java.io.IOException;  /\*\*  \* **@ClassName**: FileWriterDemo2  \* **@Description**: FileWriter注意事项  \* **@date** 2017年10月14日 下午1:30:07  \* Company www.igeekhome.com  \*  \* 输出流写数据的步骤：  \* A:创建输出流对象  \* B:调用输出流对象的写数据方法，并刷新缓冲区  \* C:释放资源  \*  \* 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** FileWriterDemo2 {  **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其它写方法

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

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

#### 案例代码三

|  |
| --- |
| **package** com.igeek\_02;  **import** java.io.FileWriter;  **import** java.io.IOException;  /\*\*  \* **@ClassName**: FileWriterDemo3  \* **@Description**: FileWriter其它写方法  \* **@date** 2017年10月14日 下午1:36:18  \* Company www.igeekhome.com  \*  \* 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  \* 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  \*/  **public** **class** FileWriterDemo3 {  **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  //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  //fw.write(chs,0,5);  fw.write(chs,2,3);    //释放资源  fw.close();  }  } |

### FileWriter写入换行以及向文本末尾追加

#### 案例代码四

|  |
| --- |
| **package** com.igeek\_02;  **import** java.io.FileWriter;  **import** java.io.IOException;  /\*\*  \* **@ClassName**: FileWriterDemo4  \* **@Description**: FileWriter写入换行以及向文本末尾追加  \* **@date** 2017年10月14日 下午1:41:34  \* Company www.igeekhome.com  \*  \* 如何实现数据的换行?  \* \n可以实现换行，但是windows系统自带的记事本打开并没有换行，这是为什么呢?  \* 因为windows识别的换行不是\n，而是\r\n  \* windows:\r\n  \* linux:\n  \* mac:\r  \* 如何实现数据的追加写入?  \* FileWriter(String fileName, boolean append)  \*/  **public** **class** FileWriterDemo4 {  **public** **static** **void** main(String[] args) **throws** IOException {  //创建输出流对象  //FileWriter fw = new FileWriter("c.txt");  FileWriter fw = **new** FileWriter("c.txt",**true**); //表示追加写入，默认是false    **for**(**int** x=0; x<10; x++) {  fw.write("hello"+x);  fw.write("\r\n");  }    //释放资源  fw.close();  }  } |

# FileReader类使用

## FileReader从文件中读数据

FileReader方法:

构造方法

FileReader(String fileName) 传入一个文件的路径

成员方法

int read() 读取单个字符

int read(char[] cbuf) 读取字符，并将字符读入数组

### FileReader读数据一次读取一个字符

#### 案例代码五

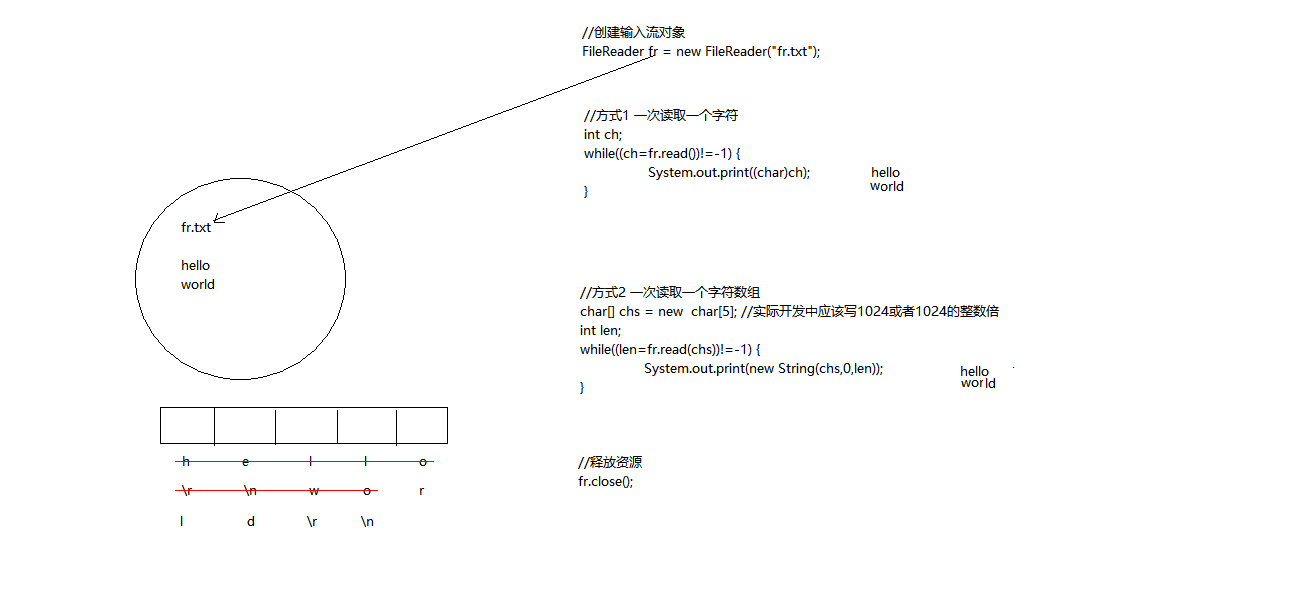
|  |
| --- |
| **package** com.igeek\_01;  **import** java.io.FileReader;  **import** java.io.IOException;  /\*\*  \* **@ClassName**: FileReaderDemo  \* **@Description**: 从文件中读数据并显示到控制台  \* **@date** 2017年10月14日 下午1:45:52  \* Company www.igeekhome.com  \*  \* 需求：从文件中读数据并显示到控制台  \* 读数据--输入流--FileReader  \*  \* FileReader:  \* FileReader(String fileName):传递文件名称  \*  \* 输入流读文件的步骤：  \* A:创建输入流对象  \* B:调用输入流对象的读数据方法  \* C:释放资源  \*  \* java.io.FileNotFoundException: fr.txt (系统找不到指定的文件。)  \*/  **public** **class** FileReaderDemo {  **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("FileWriterDemo.java");    //调用输入流对象的读数据方法  //int read():一次读取一个字符    /\*  //第一次读数据  int ch = fr.read();  System.out.println(ch);  System.out.println((char)ch);    //第二次读数据  ch = fr.read();  System.out.println(ch);  System.out.println((char)ch);    //第三次读数据  ch = fr.read();  System.out.println(ch);  System.out.println((char)ch);    //这是时候，我们发现代码的重复度很高，想用循环改进，但是不知道循环的结束条件是什么  ch = fr.read();  System.out.println(ch);    ch = fr.read();  System.out.println(ch);  //通过测试，我们知道，如果读取数据的返回值是-1的时候，就说明没有数据了，这也是我们循环的结束条件  \*/    **int** ch;  //1:fr.read()  //2:ch=fr.read()  //3:ch != -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();  }  } |

### FileReader读数据一次读取一个字符数组的数据

#### 案例代码六

|  |
| --- |
| **package** com.igeek\_01;  **import** java.io.FileReader;  **import** java.io.IOException;  /\*\*  \* **@ClassName**: FileReaderDemo2  \* **@Description**: FileReader读数据一次读取一个字符数组的数据  \* **@date** 2017年10月14日 下午7:36:42  \* Company www.igeekhome.com  \*  \* 输入流读文件的步骤：  \* A：创建输入流对象  \* B：调用输入流对象的读数据方法  \* C：释放资源  \*/  **public** **class** FileReaderDemo2 {  **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("FileReaderDemo.java");    //调用输入流对象的读数据方法    /\*  char[] chs=new char[5];  //第一次读取  int len=fr.read(chs);  System.out.println(len);  //System.out.println(chs);  System.out.println(new String(chs, 0, len));    //第二次读取  len=fr.read(chs);  System.out.println(len);  //System.out.println(chs);  System.out.println(new String(chs, 0, len));    //第三次读取  len=fr.read(chs);  System.out.println(len);  //System.out.println(chs);  System.out.println(new String(chs, 0, len));    //我们发现读取代码的方式相似度很高，想用循环改进，但是不知道什么时候会结束循环  len=fr.read(chs);  System.out.println(len);  len=fr.read(chs);  System.out.println(len);  //通过测试，我们知道，如果实际读取长度是-1的时候，说明没有数据了  \*/    **char**[] chs=**new** **char**[5];  **int** len;  //A:fr.read(chs)  //B:len=fr.read(chs)  //C:len!=-1  **while**((len=fr.read(chs))!=-1){  //System.out.println(new String(chs,0,len));  System.***out***.print(**new** String(chs,0,len));  }    //释放资源  fr.close();  }  } |

### 读取数据两种方式的图解



## 利用FileReader和FileWriter完成文件复制

### 读一次写一次

#### 案例代码七

|  |
| --- |
| **package** com.igeek\_02;  **import** java.io.FileReader;  **import** java.io.FileWriter;  **import** java.io.IOException;  /\*\*  \* **@ClassName**: CopyFileDemo  \* **@Description**: 文件复制拷贝  \* **@date** 2017年10月14日 下午1:48:48  \* Company www.igeekhome.com  \*  \* 需求：  \* 把项目路径下的FileWriterDemo.java中的内容复制到项目路径下的Copy.java中  \*  \* 文件复制也是有规律可循的，也就是说有套路。  \*  \* 数据源：  \* FileWriterDemo.java -- 读数据 -- FileReader  \* 目的地：  \* Copy.java -- 写数据 -- FileWriter  \*/  **public** **class** CopyFileDemo {  **public** **static** **void** main(String[] args) **throws** IOException {  //创建输入流对象  FileReader fr = **new** FileReader("FileWriterDemo.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();  }  } |

### 利用字符数组拷贝文件

#### 案例代码八

|  |
| --- |
| **package** com.igeek\_02;  **import** java.io.FileReader;  **import** java.io.FileWriter;  **import** java.io.IOException;  /\*\*  \* **@ClassName**: CopyFileDemo2  \* **@Description**: 利用字符数组拷贝文件  \* **@date** 2017年10月14日 下午1:51:16  \* Company www.igeekhome.com  \*  \* 需求：  \* 把项目路径下的FileWriterDemo.java中的内容复制到项目路径下的Copy.java中  \*  \* 数据源：  \* FileWriterDemo.java -- 读数据 -- FileReader  \* 目的地：  \* Copy.java -- 写数据 -- FileWriter  \*/  **public** **class** CopyFileDemo2 {  **public** **static** **void** main(String[] args) **throws** IOException {  //创建输入流对象  FileReader fr = **new** FileReader("FileWriterDemo.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();  }  } |

# 缓冲流介绍和使用

## 缓冲流的基本使用

### 案例代码九

|  |
| --- |
| **package** com.igeek\_01;  **import** java.io.BufferedReader;  **import** java.io.FileReader;  **import** java.io.IOException;  /\*\*  \* **@ClassName**: BufferedStreamDemo  \* **@Description**: 缓冲流BufferedWriter、BufferedReader的基本使用  \* **@date** 2017年10月14日 下午2:06:41  \* Company www.igeekhome.com  \*  \* 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** BufferedStreamDemo {  **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("FileWriterDemo.java"));    /\*  //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

利用缓冲流把项目路径下的FileWriterDemo.java中的内容复制到项目路径下的Copy.java中

第一种方式:使用缓冲流不使用字符数组

第二种方式:使用缓冲流使用字符数组

### 案例代码十

|  |
| --- |
| **package** com.igeek\_02;  **import** java.io.BufferedReader;  **import** java.io.BufferedWriter;  **import** java.io.FileReader;  **import** java.io.FileWriter;  **import** java.io.IOException;  /\*\*  \* **@ClassName**: CopyFileDemo  \* **@Description**: 缓冲流复制文本文件的两种方式  \* **@date** 2017年10月14日 下午2:09:54  \* Company www.igeekhome.com  \*  \* 需求：  \* 把项目路径下的FileWriterDemo.java中的内容复制到项目路径下的Copy.java中  \*  \* 数据源：  \* FileWriterDemo.java -- 读数据 -- FileReader -- 高效的读数据 -- BufferedReader  \* 目的地：  \* Copy.java -- 写数据 -- FileWriter -- 高效的写数据 -- BufferedWriter  \*/  **public** **class** CopyFileDemo {  **public** **static** **void** main(String[] args) **throws** IOException {  //创建输入缓冲流对象  BufferedReader br = **new** BufferedReader(**new** FileReader("FileWriterDemo.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();  }  } |

## 缓冲流的特有方法使用

BufferedWriter

void newLine():写一个换行符，这个换行符由系统决定,不同的操作系统newLine()方法使用的换行符不同

windows:\r\n

linux:\n

mac:\r

BufferedReader

String readLine():一次读取一行数据，但是不读取换行符

### 案例代码十一

|  |
| --- |
| **package** com.igeek\_03;  **import** java.io.BufferedReader;  **import** java.io.FileReader;  **import** java.io.IOException;  /\*\*  \* **@ClassName**: BufferedStreamDemo  \* **@Description**: 缓冲流的特殊功能  \* **@date** 2017年10月14日 下午2:12:49  \* Company www.igeekhome.com  \*  \* 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** BufferedStreamDemo {  **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 = br.readLine();  System.out.println(line);    line = br.readLine();  System.out.println(line);    line = br.readLine();  System.out.println(line);    line = br.readLine();  System.out.println(line);  \*/    String line;  //1:br.readLine()  //2:line=br.readLine()  //3:line != null  **while**((line=br.readLine())!=**null**) {  System.***out***.println(line);  }    br.close();  }  } |

## 缓冲流的特有方法复制文件

### 案例代码十二

|  |
| --- |
| **package** com.igeek\_04;  **import** java.io.BufferedReader;  **import** java.io.BufferedWriter;  **import** java.io.FileReader;  **import** java.io.FileWriter;  **import** java.io.IOException;  /\*\*  \* **@ClassName**: CopyFileDemo  \* **@Description**: 缓冲流的特有方法复制文件  \* **@date** 2017年10月14日 下午2:15:30  \* Company www.igeekhome.com  \*  \* 需求：  \* 把项目路径下的FileWriterDemo.java中的内容复制到项目路径下的Copy.java中  \*  \* 数据源：  \* FileWriterDemo.java -- 读数据 -- FileReader -- 高效的读数据 -- BufferedReader  \* 目的地：  \* Copy.java -- 写数据 -- FileWriter -- 高效的写数据 -- BufferedWriter  \*/  **public** **class** CopyFileDemo {  **public** **static** **void** main(String[] args) **throws** IOException {  //创建输入缓冲流对象  BufferedReader br = **new** BufferedReader(**new** FileReader("FileWriterDemo.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流相关案例

## 复制文本文件的5种方式

A:利用基本流一次读写一个字符

B:利用基本流一次读写一个字符数组

C:利用缓冲流一次读写一个字符

D:利用缓冲流一次读写一个字符数组

E:利用缓冲流的特有方法一次读写一个字符串

### 案例代码十三

|  |
| --- |
| **package** com.igeek\_01;  **import** java.io.BufferedReader;  **import** java.io.BufferedWriter;  **import** java.io.FileReader;  **import** java.io.FileWriter;  **import** java.io.IOException;  /\*\*  \* **@ClassName**: CopyFileTest  \* **@Description**: 复制文本文件(5种方式)  \* **@date** 2017年10月14日 下午2:18:52  \* Company www.igeekhome.com  \*  \* 复制文本文件(5种方式)  \*  \* 数据源：  \* FileWriterDemo.java  \* 目的地：  \* Copy.java  \*/  **public** **class** CopyFileTest {  **public** **static** **void** main(String[] args) **throws** IOException {  /\*  method1("FileWriterDemo.java","Copy.java");  method2("FileWriterDemo.java","Copy.java");  method3("FileWriterDemo.java","Copy.java");  method4("FileWriterDemo.java","Copy.java");  method5("FileWriterDemo.java","Copy.java");  \*/    String srcFileName = "FileWriterDemo.java";  String destFileName = "Copy.java";    // method1(srcFileName,destFileName);  // method2(srcFileName,destFileName);  *method3*(srcFileName,destFileName);  // method4(srcFileName,destFileName);  // method5(srcFileName,destFileName);  }    /\*\*  \* **@Title**: method5  \* **@Description**: 缓冲流一次读写一个字符串  \* **@param** srcFileName  \* **@param** destFileName  \* **@throws** IOException  \*/  **public** **static** **void** method5(String srcFileName,String destFileName) **throws** IOException {  //创建输入缓冲流对象  BufferedReader br = **new** BufferedReader(**new** FileReader(srcFileName));  //创建输出缓冲流对象  BufferedWriter bw = **new** BufferedWriter(**new** FileWriter(destFileName));    //一次读写一个字符串  String line;  **while**((line=br.readLine())!=**null**){  bw.write(line);  bw.newLine();  bw.flush();  }    //释放资源  bw.close();  br.close();  }    /\*\*  \* **@Title**: method4  \* **@Description**: 缓冲流一次读写一个字符数组  \* **@param** srcFileName  \* **@param** destFileName  \* **@throws** IOException  \*/  **public** **static** **void** method4(String srcFileName,String destFileName) **throws** IOException {  //创建输入缓冲流对象  BufferedReader br = **new** BufferedReader(**new** FileReader(srcFileName));  //创建输出缓冲流对象  BufferedWriter bw = **new** BufferedWriter(**new** FileWriter(destFileName));    //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();  }    /\*\*  \* **@Title**: method3  \* **@Description**: 缓冲流一次读写一个字符  \* **@param** srcFileName  \* **@param** destFileName  \* **@throws** IOException  \*/  **public** **static** **void** method3(String srcFileName,String destFileName) **throws** IOException {  //创建输入缓冲流对象  BufferedReader br = **new** BufferedReader(**new** FileReader(srcFileName));  //创建输出缓冲流对象  BufferedWriter bw = **new** BufferedWriter(**new** FileWriter(destFileName));    //一次读写一个字符  **int** ch;  **while**((ch=br.read())!=-1) {  bw.write(ch);  }    //释放资源  bw.close();  br.close();  }    /\*\*  \* **@Title**: method2  \* **@Description**: 基本流一次读写一个字符数组  \* **@param** srcFileName  \* **@param** destFileName  \* **@throws** IOException  \*/  **public** **static** **void** method2(String srcFileName,String destFileName) **throws** IOException {  //创建输入流对象  FileReader fr = **new** FileReader(srcFileName);  //创建输出流对象  FileWriter fw = **new** FileWriter(destFileName);    //一次读写一个字符数组  **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();  }    /\*\*  \* **@Title**: method1  \* **@Description**: 基本流一次读写一个字符  \* **@param** srcFileName  \* **@param** destFileName  \* **@throws** IOException  \*/  **public** **static** **void** method1(String srcFileName,String destFileName) **throws** IOException {  //创建输入流对象  FileReader fr = **new** FileReader(srcFileName);  //创建输出流对象  FileWriter fw = **new** FileWriter(destFileName);    //一次读写一个字符  **int** ch;  **while**((ch=fr.read())!=-1) {  fw.write(ch);  }    //释放资源  fw.close();  fr.close();  }  } |

## 把集合中的数据写到文本文件

把ArrayList集合中的字符串数据存储到文本文件项目根目下的array.txt中

每一个字符串元素作为文件中的一行数据

### 案例代码十四

|  |
| --- |
| **package** com.igeek\_02;  **import** java.io.BufferedWriter;  **import** java.io.FileWriter;  **import** java.io.IOException;  **import** java.util.ArrayList;  /\*\*  \* **@ClassName**: ArrayListToFileTest  \* **@Description**: 把ArrayList集合中的字符串数据存储到文本文件  \* **@date** 2017年10月14日 下午2:25:11  \* Company www.igeekhome.com  \*  \* 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  \*  \* 分析：  \* A:创建集合对象  \* B:往集合中添加字符串元素  \* C:创建输出缓冲流对象  \* D: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  \* E:释放资源  \*/  **public** **class** ArrayListToFileTest {  **public** **static** **void** main(String[] args) **throws** IOException {  //创建集合对象  ArrayList<String> array = **new** ArrayList<String>();    //往集合中添加字符串元素  array.add("hello");  array.add("world");  array.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** x=0; x<array.size(); x++) {  String s = array.get(x);  bw.write(s);  bw.newLine();  bw.flush();  }    //释放资源  bw.close();  }  } |

## 把文本文件中的数据读取到集合

从项目根目录下的array.txt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

### 案例代码十五

|  |
| --- |
| **package** com.igeek\_03;  **import** java.io.BufferedReader;  **import** java.io.FileReader;  **import** java.io.IOException;  **import** java.util.ArrayList;  /\*\*  \* **@ClassName**: FileToArrayListTest  \* **@Description**: 从文本文件中读取数据到ArrayList集合中，并遍历集合  \* **@date** 2017年10月14日 下午2:29:38  \* Company www.igeekhome.com  \*  \* 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  \*  \* 分析：  \* A:创建输入缓冲流对象  \* B:创建集合对象  \* C: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  \* D:释放资源  \* E:遍历集合  \*/  **public** **class** FileToArrayListTest {  **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> array = **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**) {  array.add(line);  }    //释放资源  br.close();    //遍历集合  **for**(**int** x=0; x<array.size(); x++) {  String s = array.get(x);  System.***out***.println(s);  }  }  } |

重点和总结

1、IO流概述及FileWriter类使用

2、FileReader类使用

3、缓冲流介绍和使用