第10天 IO

今日内容介绍

* IO流概述及FileWriter类使用
* FileReader类使用
* 缓冲流介绍和使用
* IO流相关案例

# IO流概述及FileWriter类使用

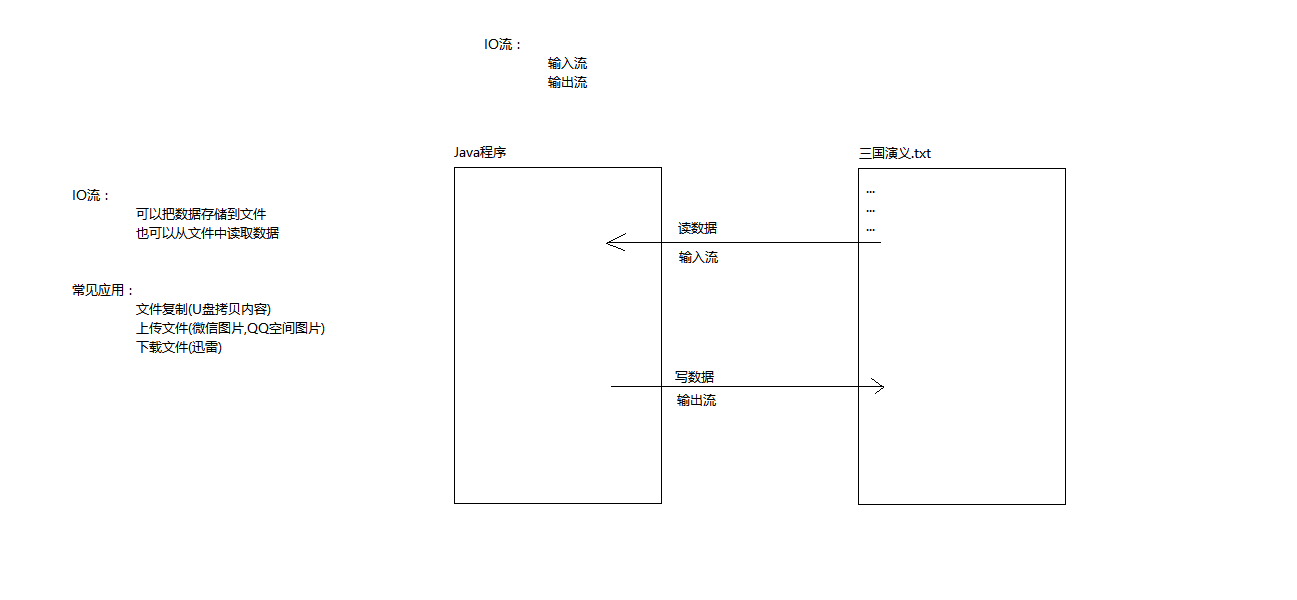
## IO流概述及分类

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

A:打开帮助文档

B:点击显示，找到索引，看到输入框

C:你要学习什么内容，你就在框框里面输入什么内容

举例：Random

D:看包

java.lang包下的类在使用的时候是不需要导包的

E:看类的描述

Random类是用于生成随机数的类

F:看构造方法

Random():无参构造方法

Random r = new Random();

G:看成员方法

public int nextInt(int n):产生的是一个[0,n)范围内的随机数

调用方法：

看返回值类型：人家返回什么类型，你就用什么类型接收

看方法名：名字不要写错了

看形式参数：人家要几个参数，你就给几个，人家要什么数据类型的，你就给什么数据类型的

int number = r.nextInt(100);

### 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.itheima\_01;

**import** java.io.FileWriter;

**import** java.io.IOException;

/\*

\* 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.itheima\_01;

**import** java.io.FileWriter;

**import** java.io.IOException;

/\*

\* 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.itheima\_01;

**import** java.io.FileWriter;

**import** java.io.IOException;

/\*

\* 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.itheima\_01;

**import** java.io.FileWriter;

**import** java.io.IOException;

/\*

\* 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类使用

# FileReader类使用

## FileReader读数据一次读取一个字符

### 案例代码五:

**package** com.itheima\_02;

**import** java.io.FileReader;

**import** java.io.IOException;

/\*

\* 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();

}

}

## 利用FileReader和FileWriter完成文件复制

### 读一次写一次

#### 案例代码六:

**package** com.itheima\_03;

**import** java.io.FileReader;

**import** java.io.FileWriter;

**import** java.io.IOException;

/\*

\* 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

#### 案例代码七:

**package** com.itheima\_03;

**import** java.io.FileReader;

**import** java.io.FileWriter;

**import** java.io.IOException;

/\*

\* 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

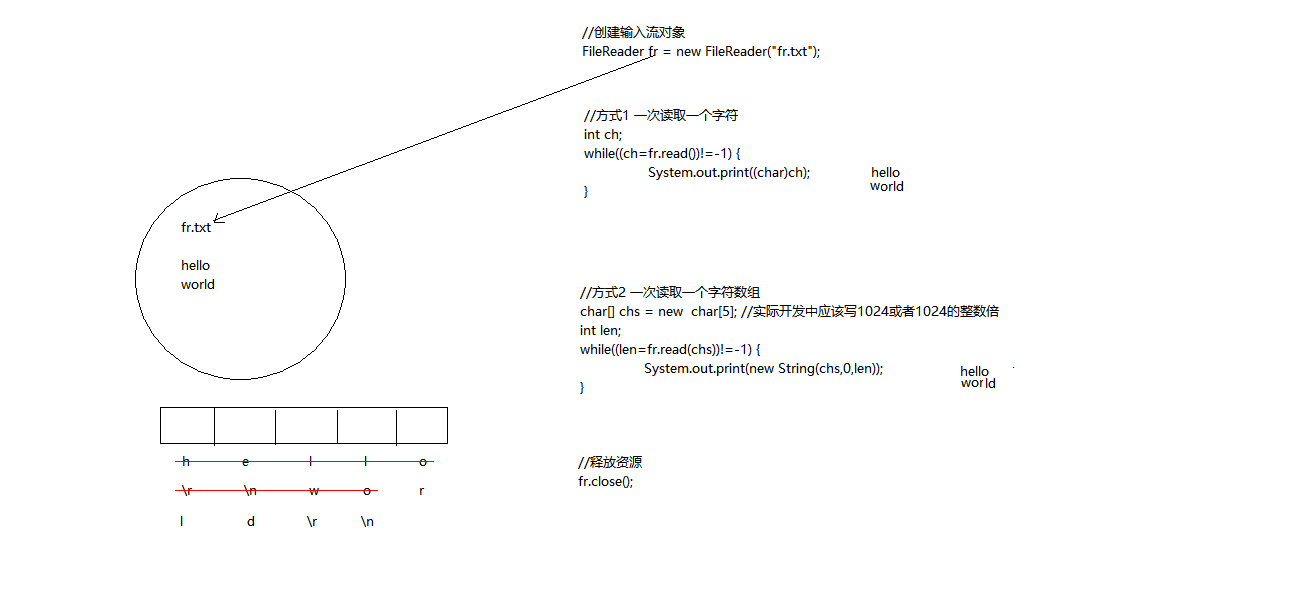
fw.close();

fr.close();

}

}

### 拷贝文件的两种方式图解



# 缓冲流介绍和使用

## 缓冲流的基本使用

### 案例代码八:

**package** com.itheima\_04;

**import** java.io.BufferedReader;

**import** java.io.FileReader;

**import** java.io.IOException;

/\*

\* 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

### 案例代码九:

**package** com.itheima\_04;

**import** java.io.BufferedReader;

**import** java.io.BufferedWriter;

**import** java.io.FileReader;

**import** java.io.FileWriter;

**import** java.io.IOException;

/\*

\* 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:

**package** com.itheima\_05;

**import** java.io.BufferedReader;

**import** java.io.FileReader;

**import** java.io.IOException;

/\*

\* 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

### 案例代码十一:

**package** com.itheima\_05;

**import** java.io.BufferedReader;

**import** java.io.BufferedWriter;

**import** java.io.FileReader;

**import** java.io.FileWriter;

**import** java.io.IOException;

/\*

\* 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

### 案例代码十二:

**package** com.itheima\_06;

**import** java.io.BufferedReader;

**import** java.io.BufferedWriter;

**import** java.io.FileReader;

**import** java.io.FileWriter;

**import** java.io.IOException;

/\*

\* 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);

}

//缓冲流一次读写一个字符串

**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();

}

//缓冲流一次读写一个字符数组

**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();

}

//缓冲流一次读写一个字符

**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();

}

//基本流一次读写一个字符数组

**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();

}

//基本流一次读写一个字符

**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

### 案例代码十三:

**package** com.itheima\_06;

**import** java.io.BufferedWriter;

**import** java.io.FileWriter;

**import** java.io.IOException;

**import** java.util.ArrayList;

/\*

\* 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

### 案例代码十四:

**package** com.itheima\_06;

**import** java.io.BufferedReader;

**import** java.io.FileReader;

**import** java.io.IOException;

**import** java.util.ArrayList;

/\*

\* 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);

}

}

}