java基础知识 第5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】对象数组**
2. 【应用】能够独立定义一个对象数组并遍历该数组
3. 【理解】能够阐述对象数组在堆中的内存划分
4. **【应用】ArrayList集合**
5. 【应用】能够独立使用集合的增删改查方法
6. 【应用】能够独立写出遍历集合代码
7. 【应用】能够完成ArrayList练习之获取满足要求的元素
8. 【应用】能够完成ArrayList练习之存储自定义对象并遍历
9. 【应用】能够完成ArrayList练习之键盘录入数据存储并遍历
10. **【应用】学生管理系统案例**
11. 【应用】能够独立完成学生管理系统之查询所有学生功能
12. 【应用】能够独立完成学生管理系统之添加不重复id的学生功能
13. 【应用】能够独立完成学生管理系统之删除学生功能
14. 【应用】能够独立完成学生管理系统之修改学生功能
15. **【应用】IO流概述及FileWriter类使用**
16. 【了解】能够阐述IO流的作用
17. 【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
18. 【应用】能够理解并用代码测试write()的重载方法
19. **【应用】FileReader类使用和缓冲流使用**
20. 【应用】能够独立使用FileReader读数据一次读取一个字符
21. 【应用】能够写出利用FileReader和FileWrtie完成文本文件复制的两种方式
22. 【应用】能够使用缓冲流BufferedReader和BufferedWriter完成文本文件复制
23. 【应用】能够使用readLine()和newLine()完成文本文件复制
24. **【应用】IO流相关案例**
25. 【应用】能够熟练写出复制文本文件的五种方式
26. 【应用】能够独立完成将集合中的数据写到文本文件中
27. 【应用】能够将文本文件的数据读取到集合中
28. **【应用】综合练习**
29. 【应用】能够独立完成把集合中的学生信息写入文本文件案例
30. 【应用】能够独立完成文件中的数据读到集合并遍历集合案例
31. 【应用】能够独立完成学生管理系统IO版案例

# 对象数组

## 对象数组概述

A:基本类型的数组:存储的元素为基本类型

int[] arr={1,2,3,4}

B:对象数组:存储的元素为引用类型

Student[] stus=new Student[3];

Student代表一个自定义类

stus数组中stus[0],stus[1],stus[2]的元素数据类型为Student,都可以指向一个Student对象

## 对象数组案例:

创建一个学生数组，存储三个学生对象并遍历

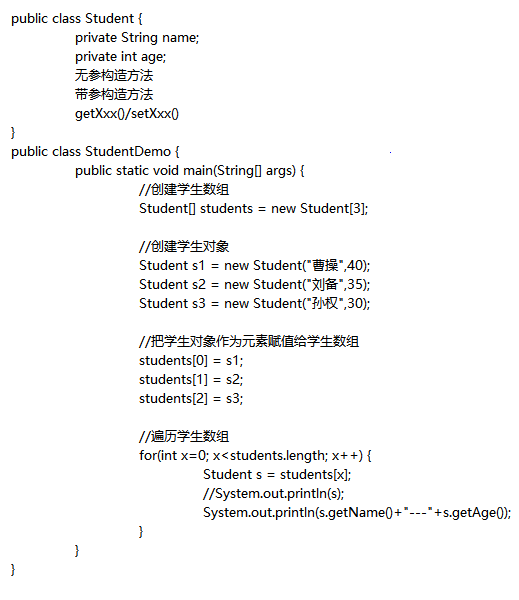
### 案例代码一

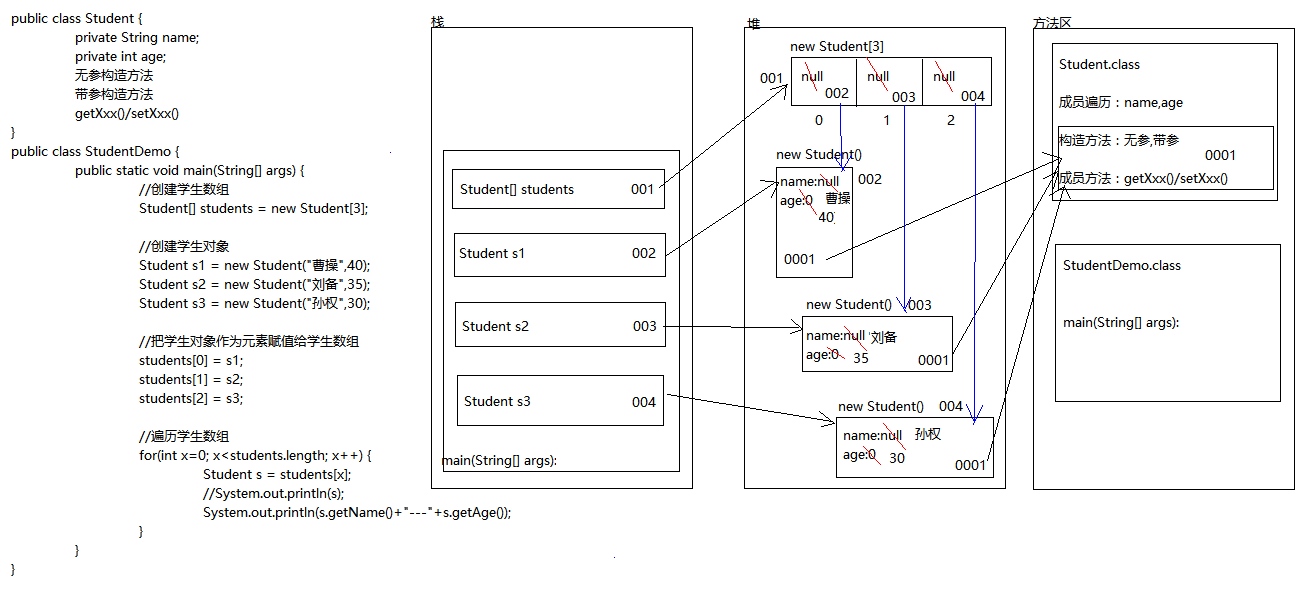
|  |
| --- |
| **package** com.igeek\_02;  /\*\*  \* **@ClassName**: Student  \* **@Description**: 学生类  \* **@date** 2017年10月14日 上午9:41:49  \* Company www.igeekhome.com  \*  \* 自动生成构造方法：  \* 代码区域右键 -- Source -- Generate Constructors from Superclass... 无参构造方法  \* 代码区域右键 -- Source -- Generate Constructor using Fields... 带参构造方法  \* 自动生成getXxx()/setXxx():  \* 代码区域右键 -- Source -- Generate Getters and Setters...  \*/  **public** **class** Student {  /\*\*  \* **@Fields** name : 姓名  \*/  **private** String name;  /\*\*  \* **@Fields** age : 年龄  \*/  **private** **int** age;    /\*\*  \* **@Title**: Student  \*/  **public** Student() {  **super**();  }    /\*\*  \* **@Title**: Student  \* **@param** name  \* **@param** age  \*/  **public** Student(String name, **int** age) {  **this**.name = name;  **this**.age = age;  }  /\*\*  \* **@return** the name  \*/  **public** String getName() {  **return** name;  }  /\*\*  \* **@param** name the name to set  \*/  **public** **void** setName(String name) {  **this**.name = name;  }  /\*\*  \* **@return** the age  \*/  **public** **int** getAge() {  **return** age;  }  /\*\*  \* **@param** age the age to set  \*/  **public** **void** setAge(**int** age) {  **this**.age = age;  }    } |

代码注释规范参照《java注释规范.docx》

|  |
| --- |
| **package** com.igeek\_02;  /\*\*  \* **@ClassName**: StudentDemo  \* **@Description**: 创建一个学生数组，存储三个学生对象并遍历  \* **@date** 2017年10月14日 上午9:55:18  \* Company www.igeekhome.com  \*  \* 创建一个学生数组，存储三个学生对象并遍历  \*  \* 分析：  \* A:定义学生类  \* B:创建学生数组  \* C:创建学生对象  \* D:把学生对象作为元素赋值给学生数组  \* E:遍历学生数组  \*/  **public** **class** StudentDemo {  **public** **static** **void** main(String[] args) {  //创建学生数组  Student[] students = **new** Student[3];    //创建学生对象  Student s1 = **new** Student("曹操",40);  Student s2 = **new** Student("刘备",35);  Student s3 = **new** Student("孙权",30);    //把学生对象作为元素赋值给学生数组  students[0] = s1;  students[1] = s2;  students[2] = s3;    //遍历学生数组  **for**(**int** x=0; x<students.length; x++) {  Student s = students[x];  //System.out.println(s);  System.***out***.println(s.getName()+"---"+s.getAge());  }  }  } |

## 对象数组的内存图





# 集合类之ArrayList

## 集合概述

A:我们学习的是面向对象编程语言，而面向对象编程语言对事物的描述都是通过对象来体现的。

为了方便对多个对象进行操作，我们就必须对这多个对象进行存储，而要想对多个对象进行存储，就不能是一个基本的变量，而应该是一个容器类型的变量。

B:到目前为止，我们学习过了哪些容器类型的数据呢？

StringBuilder,数组。

StringBuilder的结果只能是一个字符串类型，不一定满足我们的需求。

所以，我们目前只能选择数组了，也就是我们前面学习过的对象数组。

但是，数组的长度是固定的，如果有时候元素的个数不确定的,我们无 法定义出数组的长度,这个时候，java就提供了集合类供我们使用。

## ArrayList集合

### ArrayList添加新元素

#### 案例代码二

|  |
| --- |
| **package** com.igeek\_02;  **import** java.util.ArrayList;  /\*\*  \* **@ClassName**: ArrayListDemo  \* **@Description**: 集合类  \* **@date** 2017年10月14日 上午10:02:50  \* Company www.igeekhome.com  \*  \* 为什么会出现集合类：  \* 我们学习的是面向对象编程语言，而面向对象编程语言对事物的描述都是通过对象来体现的。  \* 为了方便对多个对象进行操作，我们就必须对这多个对象进行存储，而要想对多个对象进行存储，  \* 就不能是一个基本的变量，而应该是一个容器类型的变量。  \* 到目前为止，我们学习过了哪些容器类型的数据呢？StringBuilder,数组。  \* StringBuilder的结果只能是一个字符串类型，不一定满足我们的需求。  \* 所以，我们目前只能选择数组了，也就是我们前面学习过的对象数组。  \* 但是，数组的长度是固定的，适应不了变化的需求，那么，我们该如何选择呢?  \* 这个时候，java就提供了集合类供我们使用。  \*  \* 集合类的特点：  \* 长度可变。  \*  \* ArrayList<E>:  \* 大小可变数组的实现  \*  \* <E>:是一种特殊的数据类型，泛型。  \* 怎么用呢?  \* 在出现E的地方我们使用实际的引用数据类型替换即可  \* 举例：ArrayList<String>,ArrayList<Student>  \*  \* 构造方法：  \* ArrayList()  \*  \* 添加元素：  \* public boolean add(E e):添加元素  \* public void add(int index,E element):在指定的索引处添加一个元素  \*/  **public** **class** ArrayListDemo {  **public** **static** **void** main(String[] args) {  //创建集合对象  ArrayList<String> array = **new** ArrayList<String>();    //add(E e):添加元素  array.add("hello");  array.add("world");  array.add("java");    //add(int index,E element):在指定的索引处添加一个元素  //array.add(1, "android");    System.***out***.println("array:"+array);  }  } |

### ArrayList删改查方法

A:获取元素

public E get(int index):返回指定索引处的元素

B:集合长度

public int size():返回集合中的元素的个数

C:删除元素

public boolean remove(Object o):删除指定的元素，返回删除是否成功

public E remove(int index):删除指定索引处的元素，返回被删除的元素

D:修改元素

public E set(int index,E element):修改指定索引处的元素，返回被修改的元素

#### 案例代码三

|  |
| --- |
| **package** com.igeek\_02;  **import** java.util.ArrayList;  /\*\*  \* **@ClassName**: ArrayListDemo2  \* **@Description**: ArrayList的常用操作  \* **@date** 2017年10月14日 上午10:05:12  \* Company www.igeekhome.com  \*  \* 获取元素  \* public E get(int index):返回指定索引处的元素  \* 集合长度  \* public int size():返回集合中的元素的个数  \* 删除元素  \* public boolean remove(Object o):删除指定的元素，返回删除是否成功  \* public E remove(int index):删除指定索引处的元素，返回被删除的元素  \* 修改元素  \* public E set(int index,E element):修改指定索引处的元素，返回被修改的元素  \*/  **public** **class** ArrayListDemo2 {  **public** **static** **void** main(String[] args) {  //创建集合对象  ArrayList<String> array = **new** ArrayList<String>();    //添加元素  array.add("hello");  array.add("world");  array.add("java");    //public E get(int index):返回指定索引处的元素  //System.out.println("get:"+array.get(0));  //System.out.println("get:"+array.get(1));  //System.out.println("get:"+array.get(2));    //public int size():返回集合中的元素的个数  //System.out.println("size:"+array.size());    //public boolean remove(Object o):删除指定的元素，返回删除是否成功  //System.out.println("remove:"+array.remove("world"));//true  //System.out.println("remove:"+array.remove("world"));//false    //public E remove(int index):删除指定索引处的元素，返回被删除的元素  //System.out.println("remove:"+array.remove(0));    //public E set(int index,E element):修改指定索引处的元素，返回被修改的元素  System.***out***.println("set:"+array.set(1, "android"));    //输出  System.***out***.println("array:"+array);  }  } |

### ArrayList遍历

集合的遍历思想和数组的遍历思想相同

循环遍历容器,依次取出里面的元素即可

#### 案例代码四

|  |
| --- |
| **package** com.igeek\_02;  **import** java.util.ArrayList;  /\*\*  \* **@ClassName**: ArrayListDemo3  \* **@Description**: ArrayList集合的遍历  \* **@date** 2017年10月14日 上午10:07:02  \* Company www.igeekhome.com  \*  \* ArrayList集合的遍历  \* 通过size()和get()配合实现的  \*/  **public** **class** ArrayListDemo3 {  **public** **static** **void** main(String[] args) {  //创建集合对象  ArrayList<String> array = **new** ArrayList<String>();    //添加元素  array.add("hello");  array.add("world");  array.add("java");    //获取元素  //原始做法  System.***out***.println(array.get(0));  System.***out***.println(array.get(1));  System.***out***.println(array.get(2));  System.***out***.println("----------");    **for**(**int** x=0; x<3; x++) {  System.***out***.println(array.get(x));  }  System.***out***.println("----------");    //如何知道集合中元素的个数呢?size()  **for**(**int** x=0; x<array.size(); x++) {  System.***out***.println(array.get(x));  }  System.***out***.println("----------");    //最标准的用法  **for**(**int** x=0; x<array.size(); x++) {  String s = array.get(x);  System.***out***.println(s);  }  }  } |

## ArrayList集合案例

### ArrayList练习之存储字符串并遍历

向集合中添加任意四个字符串,遍历集合,依次打印取出的字符串

#### 案例代码五

|  |
| --- |
| **package** com.igeek\_03;  **import** java.util.ArrayList;  /\*\*  \* **@ClassName**: ArrayListTest  \* **@Description**: 存储字符串并遍历  \* **@date** 2017年10月14日 上午10:09:10  \* Company www.igeekhome.com  \*  \* 存储字符串并遍历  \*  \* 分析：  \* A:创建集合对象  \* B:添加字符串元素  \* C:遍历集合  \*/  **public** **class** ArrayListTest {  **public** **static** **void** main(String[] args) {  //创建集合对象  ArrayList<String> array = **new** ArrayList<String>();    //添加字符串元素  array.add("向问天");  array.add("刘正风");  array.add("左冷禅");  array.add("风清扬");    //遍历集合  **for**(**int** x=0; x<array.size(); x++) {  String s = array.get(x);  System.***out***.println(s);  }  }  } |

### ArrayList练习之获取满足要求的元素

给定一个字符串数组：{“张三丰”,“宋远桥”,“张无忌”,“殷梨亭”,“张翠山”,“莫声谷”}，将数组中的元素添加到集合中，并把所有姓张的人员打印到控制台上

#### 案例代码六

|  |
| --- |
| **package** com.igeek\_03;  **import** java.util.ArrayList;  /\*\*  \* **@ClassName**: ArrayListTest2  \* **@Description**: ArrayList练习之获取满足要求的元素  \* **@date** 2017年10月14日 上午10:11:10  \* Company www.igeekhome.com  \*  \* 给定一个字符串数组：{“张三丰”,“宋远桥”,“张无忌”,“殷梨亭”,“张翠山”,“莫声谷”}，  \* 将数组中的元素添加到集合中，并把所有姓张的人员打印到控制台上。  \*  \* 分析：  \* A:定义字符串数组  \* B:创建集合对象  \* C:遍历字符串数组，获取到每一个字符串元素  \* D:把获取到的字符串元素添加到集合  \* E:遍历集合  \* 要判断每一个字符串元素是否以"张"开头，如果是，就输出在控制台  \*/  **public** **class** ArrayListTest2 {  **public** **static** **void** main(String[] args) {  //定义字符串数组  String[] strArray = {"张三丰","宋远桥","张无忌","殷梨亭","张翠山","莫声谷"};    //创建集合对象  ArrayList<String> array = **new** ArrayList<String>();    //遍历字符串数组，获取到每一个字符串元素  **for**(**int** x=0; x<strArray.length; x++) {  //把获取到的字符串元素添加到集合  array.add(strArray[x]);  }    //遍历集合  **for**(**int** x=0; x<array.size(); x++) {  String s = array.get(x);  //要判断每一个字符串元素是否以"张"开头，如果是，就输出在控制台  **if**(s.startsWith("张")) {  System.***out***.println(s);  }  }  }  } |

### ArrayList练习之存储自定义对象并遍历

A:自定义一个学生类,学生中有姓名和年龄属性,生成满参构造与空参构造

生成属性对应的getter/setter方法

B:在测试类中使用满参构造创建三个学生对象,然后将每个学生对象均添加到ArrayList集合中

C:遍历这个ArrayList集合,依次打印出每个学生的姓名和年龄

#### 案例代码七

|  |
| --- |
| **package** com.igeek\_03;  /\*\*  \* **@ClassName**: Student  \* **@Description**: 学生类  \* **@date** 2017年10月14日 上午9:41:49  \* Company www.igeekhome.com  \*  \* 自动生成构造方法：  \* 代码区域右键 -- Source -- Generate Constructors from Superclass... 无参构造方法  \* 代码区域右键 -- Source -- Generate Constructor using Fields... 带参构造方法  \* 自动生成getXxx()/setXxx():  \* 代码区域右键 -- Source -- Generate Getters and Setters...  \*/  **public** **class** Student {  /\*\*  \* **@Fields** name : 姓名  \*/  **private** String name;  /\*\*  \* **@Fields** age : 年龄  \*/  **private** **int** age;    /\*\*  \* **@Title**: Student  \*/  **public** Student() {  **super**();  }    /\*\*  \* **@Title**: Student  \* **@param** name  \* **@param** age  \*/  **public** Student(String name, **int** age) {  **this**.name = name;  **this**.age = age;  }  /\*\*  \* **@return** the name  \*/  **public** String getName() {  **return** name;  }  /\*\*  \* **@param** name the name to set  \*/  **public** **void** setName(String name) {  **this**.name = name;  }  /\*\*  \* **@return** the age  \*/  **public** **int** getAge() {  **return** age;  }  /\*\*  \* **@param** age the age to set  \*/  **public** **void** setAge(**int** age) {  **this**.age = age;  }    } |

|  |
| --- |
| **package** com.igeek\_03;  **import** java.util.ArrayList;  /\*\*  \* **@ClassName**: ArrayListTest3  \* **@Description**: 存储自定义对象并遍历  \* **@date** 2017年10月14日 上午10:14:21  \* Company www.igeekhome.com  \*  \* 存储自定义对象并遍历  \*  \* 分析：  \* A:定义学生类  \* B:创建集合对象  \* C:创建学生对象  \* D:把学生对象作为元素添加到集合中  \* E:遍历集合  \*/  **public** **class** ArrayListTest3 {  **public** **static** **void** main(String[] args) {  //创建集合对象  ArrayList<Student> array = **new** ArrayList<Student>();    //创建学生对象  Student s1 = **new** Student("林青霞",28);  Student s2 = **new** Student("张曼玉",30);  Student s3 = **new** Student("景甜",25);  Student s4 = **new** Student("柳岩",18);    //把学生对象作为元素添加到集合中  array.add(s1);  array.add(s2);  array.add(s3);  array.add(s4);    //遍历集合  **for**(**int** x=0; x<array.size(); x++) {  Student s = array.get(x);  System.***out***.println(s.getName()+"---"+s.getAge());  }  }  } |

### ArrayList练习之键盘录入数据存储并遍历

创建一个Student类包含姓名和年龄属性

创建一个ArrayList集合

向集合中添加三个Student对象Student对象中姓名和年龄的数据均来自与键盘录入

最终遍历这个集合,取出Student对象以及里面属性的值

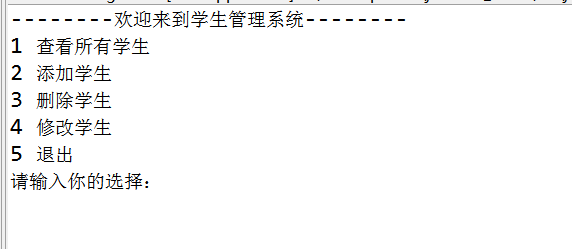
#### 案例代码八

|  |
| --- |
| **package** com.igeek\_03\_04;  /\*\*  \* **@ClassName**: Student  \* **@Description**: 学生类  \* **@date** 2017年10月14日 上午9:41:49  \* Company www.igeekhome.com  \*  \* 自动生成构造方法：  \* 代码区域右键 -- Source -- Generate Constructors from Superclass... 无参构造方法  \* 代码区域右键 -- Source -- Generate Constructor using Fields... 带参构造方法  \* 自动生成getXxx()/setXxx():  \* 代码区域右键 -- Source -- Generate Getters and Setters...  \*/  **public** **class** Student {  /\*\*  \* **@Fields** name : 姓名  \*/  **private** String name;  /\*\*  \* **@Fields** age : 年龄  \*/  **private** String age;    /\*\*  \* **@Title**: Student  \*/  **public** Student() {  **super**();  }    /\*\*  \* **@Title**: Student  \* **@param** name  \* **@param** age  \*/  **public** Student(String name, String age) {  **this**.name = name;  **this**.age = age;  }  /\*\*  \* **@return** the name  \*/  **public** String getName() {  **return** name;  }  /\*\*  \* **@param** name the name to set  \*/  **public** **void** setName(String name) {  **this**.name = name;  }  /\*\*  \* **@return** the age  \*/  **public** String getAge() {  **return** age;  }  /\*\*  \* **@param** age the age to set  \*/  **public** **void** setAge(String age) {  **this**.age = age;  }    } |

|  |
| --- |
| **package** com.igeek\_03\_04;  **import** java.util.ArrayList;  **import** java.util.Scanner;  /\*\*  \* **@ClassName**: StudentDemo  \* **@Description**: ArrayList练习之键盘录入数据存储并遍历  \* **@date** 2017年10月14日 上午10:22:12  \* Company www.igeekhome.com  \*  \* 创建一个集合，存储学生对象，学生对象的数据来自于键盘录入，最后，遍历集合  \*  \* 注意：为了方便使用，我把学生类中的所有成员定义为String类型  \*  \* 分析：  \* A:定义学生类  \* B:创建集合对象  \* C:键盘录入数据,创建学生对象,把键盘录入的数据赋值给学生对象的成员变量  \* D:把学生对象作为元素存储到集合中  \* E:遍历集合  \*  \*/  **public** **class** StudentDemo {  **public** **static** **void** main(String[] args) {  //创建集合对象  ArrayList<Student> array = **new** ArrayList<Student>();    /\*  //键盘录入数据,创建学生对象,把键盘录入的数据赋值给学生对象的成员变量  Scanner sc = new Scanner(System.in);  System.out.println("请输入学生姓名：");  String name = sc.nextLine();  System.out.println("请输入学生年龄：");  String age = sc.nextLine();    Student s = new Student();  s.setName(name);  s.setAge(age);    //把学生对象作为元素存储到集合中  array.add(s);  \*/    //为了提高代码的复用性，我把键盘录入数据给学生对象，并存储到集合中的动作用一个方法来实现    //调用方法  *addStudent*(array);  *addStudent*(array);  *addStudent*(array);    //遍历集合  **for**(**int** x=0; x<array.size(); x++) {  Student s = array.get(x);  System.***out***.println(s.getName()+"---"+s.getAge());  }  }    /\*\*  \* **@Title**: addStudent  \* **@Description**: 添加新的学生  \* **@param** array  \* 两个明确：  \* 返回值类型：void  \* 参数列表：ArrayList<Student> array  \*/  **public** **static** **void** addStudent(ArrayList<Student> array) {  //键盘录入数据,创建学生对象,把键盘录入的数据赋值给学生对象的成员变量  Scanner sc = **new** Scanner(System.***in***);  System.***out***.println("请输入学生姓名：");  String name = sc.nextLine();  System.***out***.println("请输入学生年龄：");  String age = sc.nextLine();    Student s = **new** Student();  s.setName(name);  s.setAge(age);    //把学生对象作为元素存储到集合中  array.add(s);  }  } |

# 学生管理系统案例

## 学生管理系统案例需求



利用集合完成对学生的增删改查四个功能

## 学生管理系统案例实现

### 创建学生类

#### 案例代码九

|  |
| --- |
| **package** com.igeek\_02\_01;  /\*\*  \* **@ClassName**: Student  \* **@Description**: 学生类  \* **@date** 2017年10月14日 上午10:35:59  \* Company www.igeekhome.com  \*  \*/  **public** **class** Student {  /\*\*  \* **@Fields** id : 学号  \*/  **private** String id;  /\*\*  \* **@Fields** name : 姓名  \*/  **private** String name;  /\*\*  \* **@Fields** age : 年龄  \*/  **private** String age;  /\*\*  \* **@Fields** address : 居住地  \*/  **private** String address;  /\*\*  \* **@Title**: Student  \*/  **public** Student() {  }  /\*\*  \* **@Title**: Student  \* **@param** id  \* **@param** name  \* **@param** age  \* **@param** address  \*/  **public** Student(String id, String name, String age, String address) {  **this**.id = id;  **this**.name = name;  **this**.age = age;  **this**.address = address;  }  /\*\*  \* **@return** the id  \*/  **public** String getId() {  **return** id;  }  /\*\*  \* **@param** id the id to set  \*/  **public** **void** setId(String id) {  **this**.id = id;  }  /\*\*  \* **@return** the name  \*/  **public** String getName() {  **return** name;  }  /\*\*  \* **@param** name the name to set  \*/  **public** **void** setName(String name) {  **this**.name = name;  }  /\*\*  \* **@return** the age  \*/  **public** String getAge() {  **return** age;  }  /\*\*  \* **@param** age the age to set  \*/  **public** **void** setAge(String age) {  **this**.age = age;  }  /\*\*  \* **@return** the address  \*/  **public** String getAddress() {  **return** address;  }  /\*\*  \* **@param** address the address to set  \*/  **public** **void** setAddress(String address) {  **this**.address = address;  }  } |

|  |
| --- |
| **package** com.igeek\_02\_01;  /\*\*  \* **@ClassName**: StudentManagerTest  \* **@Description**: 学生管理系统的主类  \* **@date** 2017年10月14日 上午10:35:59  \* Company www.igeekhome.com  \*  \* 这是我的学生管理系统的主类  \*  \* 步骤如下：  \* A：定义学生类  \* B：学生管理系统的主界面的代码编写  \* C：学生管理系统的查看所有学生的代码编写  \* D：学生管理系统的添加学生的代码编写  \* E：学生管理系统的删除学生的代码编写  \* F：学生管理系统的修改学生的代码编写  \*  \*/  **public** **class** StudentManagerTest {  } |

### 学生管理系统界面实现

#### 案例代码十

|  |
| --- |
| **package** com.igeek\_02\_02;  **import** java.util.ArrayList;  **import** java.util.Scanner;  /\*\*  \* **@ClassName**: StudentManagerTest  \* **@Description**: 学生管理系统的主类  \* **@date** 2017年10月14日 上午10:50:19  \* Company www.igeekhome.com  \*  \* 这是我的学生管理系统的主类  \*  \* 步骤如下：  \* A:定义学生类  \* B:学生管理系统的主界面的代码编写  \* C:学生管理系统的查看所有学生的代码编写  \* D:学生管理系统的添加学生的代码编写  \* E:学生管理系统的删除学生的代码编写  \* F:学生管理系统的修改学生的代码编写  \*/  **public** **class** StudentManagerTest {  **public** **static** **void** main(String[] args) {  // 为了让程序能够回到这里来，我们使用循环  **while** (**true**) {  // 这是学生管理系统的主界面  System.***out***.println("--------欢迎来到学生管理系统--------");  System.***out***.println("1 查看所有学生");  System.***out***.println("2 添加学生");  System.***out***.println("3 删除学生");  System.***out***.println("4 修改学生");  System.***out***.println("5 退出");  System.***out***.println("请输入你的选择：");  // 创建键盘录入对象  Scanner sc = **new** Scanner(System.***in***);  String choiceString = sc.nextLine();  // 用switch语句实现选择  **switch** (choiceString) {  **case** "1":  // 查看所有学生  **break**;  **case** "2":  // 添加学生  **break**;  **case** "3":  // 删除学生  **break**;  **case** "4":  // 修改学生  **break**;  **case** "5":  // 退出  // System.out.println("谢谢你的使用");  // break;  **default**:  System.***out***.println("谢谢你的使用");  System.*exit*(0); // JVM退出  **break**;  }  }  }  } |

### 学生管理系统之查询所有学生功能

#### 案例代码十一

|  |
| --- |
| **package** com.igeek\_02\_03;  **import** java.util.ArrayList;  **import** java.util.Scanner;  /\*\*  \* **@ClassName**: StudentManagerTest  \* **@Description**: 学生管理系统的主类  \* **@date** 2017年10月14日 上午10:50:19  \* Company www.igeekhome.com  \*  \* 这是我的学生管理系统的主类  \*  \* 步骤如下：  \* A:定义学生类  \* B:学生管理系统的主界面的代码编写  \* C:学生管理系统的查看所有学生的代码编写  \* D:学生管理系统的添加学生的代码编写  \* E:学生管理系统的删除学生的代码编写  \* F:学生管理系统的修改学生的代码编写  \*/  **public** **class** StudentManagerTest {  **public** **static** **void** main(String[] args) {  // 创建集合对象，用于存储学生数据  ArrayList<Student> array = **new** ArrayList<Student>();  // 为了让程序能够回到这里来，我们使用循环  **while** (**true**) {  // 这是学生管理系统的主界面  System.***out***.println("--------欢迎来到学生管理系统--------");  System.***out***.println("1 查看所有学生");  System.***out***.println("2 添加学生");  System.***out***.println("3 删除学生");  System.***out***.println("4 修改学生");  System.***out***.println("5 退出");  System.***out***.println("请输入你的选择：");  // 创建键盘录入对象  Scanner sc = **new** Scanner(System.***in***);  String choiceString = sc.nextLine();  // 用switch语句实现选择  **switch** (choiceString) {  **case** "1":  // 查看所有学生  *findAllStudent*(array);  **break**;  **case** "2":  // 添加学生  **break**;  **case** "3":  // 删除学生  **break**;  **case** "4":  // 修改学生  **break**;  **case** "5":  // 退出  // System.out.println("谢谢你的使用");  // break;  **default**:  System.***out***.println("谢谢你的使用");  System.*exit*(0); // JVM退出  **break**;  }  }  }    /\*\*  \* **@Title**: findAllStudent  \* **@Description**: 查看所有学生  \* **@param** array  \*/  **public** **static** **void** findAllStudent(ArrayList<Student> array) {  //首先来判断集合中是否有数据，如果没有数据，就给出提示，并让该方法不继续往下执行  **if**(array.size() == 0) {  System.***out***.println("不好意思,目前没有学生信息可供查询,请回去重新选择你的操作");  **return**;  }    //\t 其实就是一个tab键的位置  System.***out***.println("学号\t姓名\t年龄\t居住地");  **for**(**int** x=0; x<array.size(); x++) {  Student s = array.get(x);  System.***out***.println(s.getId()+"\t"+s.getName()+"\t"+s.getAge()+"\t"+s.getAddress());  }  }    } |

### 学生管理系统之添加学生功能

#### 案例代码十二

|  |
| --- |
| **package** com.igeek\_02\_04;  **import** java.util.ArrayList;  **import** java.util.Scanner;  /\*\*  \* **@ClassName**: StudentManagerTest  \* **@Description**: 学生管理系统的主类  \* **@date** 2017年10月14日 上午10:50:19  \* Company www.igeekhome.com  \*  \* 这是我的学生管理系统的主类  \*  \* 步骤如下：  \* A:定义学生类  \* B:学生管理系统的主界面的代码编写  \* C:学生管理系统的查看所有学生的代码编写  \* D:学生管理系统的添加学生的代码编写  \* E:学生管理系统的删除学生的代码编写  \* F:学生管理系统的修改学生的代码编写  \*/  **public** **class** StudentManagerTest {  **public** **static** **void** main(String[] args) {  // 创建集合对象，用于存储学生数据  ArrayList<Student> array = **new** ArrayList<Student>();  // 为了让程序能够回到这里来，我们使用循环  **while** (**true**) {  // 这是学生管理系统的主界面  System.***out***.println("--------欢迎来到学生管理系统--------");  System.***out***.println("1 查看所有学生");  System.***out***.println("2 添加学生");  System.***out***.println("3 删除学生");  System.***out***.println("4 修改学生");  System.***out***.println("5 退出");  System.***out***.println("请输入你的选择：");  // 创建键盘录入对象  Scanner sc = **new** Scanner(System.***in***);  String choiceString = sc.nextLine();  // 用switch语句实现选择  **switch** (choiceString) {  **case** "1":  // 查看所有学生  *findAllStudent*(array);  **break**;  **case** "2":  // 添加学生  *addStudent*(array);  **break**;  **case** "3":  // 删除学生  **break**;  **case** "4":  // 修改学生  **break**;  **case** "5":  // 退出  // System.out.println("谢谢你的使用");  // break;  **default**:  System.***out***.println("谢谢你的使用");  System.*exit*(0); // JVM退出  **break**;  }  }  }    /\*  //添加学生  public static void addStudent(ArrayList<Student> array) {  //创建键盘录入对象  Scanner sc = new Scanner(System.in);    System.out.println("请输入学生学号：");  String id = sc.nextLine();  System.out.println("请输入学生姓名：");  String name = sc.nextLine();  System.out.println("请输入学生年龄：");  String age = sc.nextLine();  System.out.println("请输入学生居住地：");  String address = sc.nextLine();    //创建学生对象  Student s = new Student();  s.setId(id);  s.setName(name);  s.setAge(age);  s.setAddress(address);    //把学生对象作为元素添加到集合  array.add(s);    //给出提示  System.out.println("添加学生成功");  }  \*/    /\*\*  \* **@Title**: addStudent  \* **@Description**: 添加学生  \* **@param** array  \*/  **public** **static** **void** addStudent(ArrayList<Student> array) {  //创建键盘录入对象  Scanner sc = **new** Scanner(System.***in***);    //为了让id能够被访问到，我们就把id定义在了循环的外面  String id;    //为了让代码能够回到这里，用循环  **while**(**true**) {  System.***out***.println("请输入学生学号：");  //String id = sc.nextLine();  id = sc.nextLine();    //判断学号有没有被人占用  //定义标记  **boolean** flag = **false**;  //遍历集合，得到每一个学生  **for**(**int** x=0; x<array.size(); x++) {  Student s = array.get(x);  //获取该学生的学号，和键盘录入的学号进行比较  **if**(s.getId().equals(id)) {  flag = **true**; //说明学号被占用了  **break**;  }  }    **if**(flag) {  System.***out***.println("你输入的学号已经被占用,请重新输入");  }**else** {  **break**; //结束循环  }  }    System.***out***.println("请输入学生姓名：");  String name = sc.nextLine();  System.***out***.println("请输入学生年龄：");  String age = sc.nextLine();  System.***out***.println("请输入学生居住地：");  String address = sc.nextLine();    //创建学生对象  Student s = **new** Student();  s.setId(id);  s.setName(name);  s.setAge(age);  s.setAddress(address);    //把学生对象作为元素添加到集合  array.add(s);    //给出提示  System.***out***.println("添加学生成功");  }  } |

### 学生管理系统之删除学生功能

#### 案例代码十三

|  |
| --- |
| **package** com.igeek\_02\_05;  **import** java.util.ArrayList;  **import** java.util.Scanner;  /\*\*  \* **@ClassName**: StudentManagerTest  \* **@Description**: 学生管理系统的主类  \* **@date** 2017年10月14日 上午10:50:19  \* Company www.igeekhome.com  \*  \* 这是我的学生管理系统的主类  \*  \* 步骤如下：  \* A:定义学生类  \* B:学生管理系统的主界面的代码编写  \* C:学生管理系统的查看所有学生的代码编写  \* D:学生管理系统的添加学生的代码编写  \* E:学生管理系统的删除学生的代码编写  \* F:学生管理系统的修改学生的代码编写  \*/  **public** **class** StudentManagerTest {  **public** **static** **void** main(String[] args) {  // 创建集合对象，用于存储学生数据  ArrayList<Student> array = **new** ArrayList<Student>();  // 为了让程序能够回到这里来，我们使用循环  **while** (**true**) {  // 这是学生管理系统的主界面  System.***out***.println("--------欢迎来到学生管理系统--------");  System.***out***.println("1 查看所有学生");  System.***out***.println("2 添加学生");  System.***out***.println("3 删除学生");  System.***out***.println("4 修改学生");  System.***out***.println("5 退出");  System.***out***.println("请输入你的选择：");  // 创建键盘录入对象  Scanner sc = **new** Scanner(System.***in***);  String choiceString = sc.nextLine();  // 用switch语句实现选择  **switch** (choiceString) {  **case** "1":  //查看所有学生  *findAllStudent*(array);  **break**;  **case** "2":  //添加学生  *addStudent*(array);  **break**;  **case** "3":  //删除学生  *deleteStudent*(array);  **break**;  **case** "4":  // 修改学生  **break**;  **case** "5":  // 退出  // System.out.println("谢谢你的使用");  // break;  **default**:  System.***out***.println("谢谢你的使用");  System.*exit*(0); // JVM退出  **break**;  }  }  }    /\*\*  \* **@Title**: deleteStudent  \* **@Description**: 删除学生  \* **@param** array  \*/  **public** **static** **void** deleteStudent(ArrayList<Student> array) {  //删除学生的思路：键盘录入一个学号，到集合中去查找，看是否有学生使用的是该学号，如果有就删除该学生  //创建键盘录入对象  Scanner sc = **new** Scanner(System.***in***);  System.***out***.println("请输入你要删除的学生的学号：");  String id = sc.nextLine();    /\*  //遍历集合  for(int x=0; x<array.size(); x++) {  //获取到每一个学生对象  Student s = array.get(x);  //拿这个学生对象的学号和键盘录入的学号进行比较  if(s.getId().equals(id)) {  array.remove(x); //根据索引删除  break;  }  }    //给出提示  System.out.println("删除学生成功");  \*/    //我们必须给出学号不存在的时候的提示    //定义一个索引  **int** index = -1;    //遍历集合  **for**(**int** x=0; x<array.size(); x++) {  //获取到每一个学生对象  Student s = array.get(x);  //拿这个学生对象的学号和键盘录入的学号进行比较  **if**(s.getId().equals(id)) {  index = x;  **break**;  }  }    **if**(index == -1) {  System.***out***.println("不好意思,你要删除的学号对应的学生信息不存在,请回去重新你的选择");  }**else** {  array.remove(index);  System.***out***.println("删除学生成功");  }    }  } |

### 学生管理系统之修改学生功能

#### 案例代码十四

|  |
| --- |
| **package** com.igeek\_02\_06;  **import** java.util.ArrayList;  **import** java.util.Scanner;  /\*\*  \* **@ClassName**: StudentManagerTest  \* **@Description**: 学生管理系统的主类  \* **@date** 2017年10月14日 上午10:50:19  \* Company www.igeekhome.com  \*  \* 这是我的学生管理系统的主类  \*  \* 步骤如下：  \* A:定义学生类  \* B:学生管理系统的主界面的代码编写  \* C:学生管理系统的查看所有学生的代码编写  \* D:学生管理系统的添加学生的代码编写  \* E:学生管理系统的删除学生的代码编写  \* F:学生管理系统的修改学生的代码编写  \*/  **public** **class** StudentManagerTest {  **public** **static** **void** main(String[] args) {  // 创建集合对象，用于存储学生数据  ArrayList<Student> array = **new** ArrayList<Student>();  // 为了让程序能够回到这里来，我们使用循环  **while** (**true**) {  // 这是学生管理系统的主界面  System.***out***.println("--------欢迎来到学生管理系统--------");  System.***out***.println("1 查看所有学生");  System.***out***.println("2 添加学生");  System.***out***.println("3 删除学生");  System.***out***.println("4 修改学生");  System.***out***.println("5 退出");  System.***out***.println("请输入你的选择：");  // 创建键盘录入对象  Scanner sc = **new** Scanner(System.***in***);  String choiceString = sc.nextLine();  // 用switch语句实现选择  **switch** (choiceString) {  **case** "1":  // 查看所有学生  *findAllStudent*(array);  **break**;  **case** "2":  //添加学生  *addStudent*(array);  **break**;  **case** "3":  //删除学生  *deleteStudent*(array);  **break**;  **case** "4":  // 修改学生  *updateStudent*(array);  **break**;  **case** "5":  // 退出  // System.out.println("谢谢你的使用");  // break;  **default**:  System.***out***.println("谢谢你的使用");  System.*exit*(0); // JVM退出  **break**;  }  }  }    /\*\*  \* **@Title**: updateStudent  \* **@Description**: 修改学生  \* **@param** array  \*/  **public** **static** **void** updateStudent(ArrayList<Student> array) {  //修改学生的思路：键盘录入一个学号，到集合中去查找，看是否有学生使用的是该学号，如果有就修改该学生  //创建键盘录入对象  Scanner sc = **new** Scanner(System.***in***);  System.***out***.println("请输入你要修改的学生的学号：");  String id = sc.nextLine();    //定义一个索引  **int** index = -1;    //遍历集合  **for**(**int** x=0; x<array.size(); x++) {  //获取每一个学生对象  Student s = array.get(x);  //拿学生对象的学号和键盘录入的学号进行比较  **if**(s.getId().equals(id)) {  index = x;  **break**;  }  }    **if**(index == -1) {  System.***out***.println("不好意思,你要修改的学号对应的学生信息不存在,请回去重新你的选择");  }**else** {  System.***out***.println("请输入学生新姓名：");  String name = sc.nextLine();  System.***out***.println("请输入学生新年龄：");  String age = sc.nextLine();  System.***out***.println("请输入学生新居住地：");  String address = sc.nextLine();    //创建学生对象  Student s = **new** Student();  s.setId(id);  s.setName(name);  s.setAge(age);  s.setAddress(address);    //修改集合中的学生对象  array.set(index, s);    //给出提示  System.***out***.println("修改学生成功");  }  }  } |

## 学生管理系统最终完整代码

### 案例完整代码十五

|  |
| --- |
| **package** com.igeek\_03;  **import** java.util.ArrayList;  **import** java.util.Scanner;  /\*\*  \* **@ClassName**: StudentManagerTest  \* **@Description**: 学生管理系统的主类  \* **@date** 2017年10月14日 上午10:50:19  \* Company www.igeekhome.com  \*  \* 这是我的学生管理系统的主类  \*  \* 步骤如下：  \* A:定义学生类  \* B:学生管理系统的主界面的代码编写  \* C:学生管理系统的查看所有学生的代码编写  \* D:学生管理系统的添加学生的代码编写  \* E:学生管理系统的删除学生的代码编写  \* F:学生管理系统的修改学生的代码编写  \*/  **public** **class** StudentManagerTest {  **public** **static** **void** main(String[] args) {  // 创建集合对象，用于存储学生数据  ArrayList<Student> array = **new** ArrayList<Student>();  // 为了让程序能够回到这里来，我们使用循环  **while** (**true**) {  // 这是学生管理系统的主界面  System.***out***.println("--------欢迎来到学生管理系统--------");  System.***out***.println("1 查看所有学生");  System.***out***.println("2 添加学生");  System.***out***.println("3 删除学生");  System.***out***.println("4 修改学生");  System.***out***.println("5 退出");  System.***out***.println("请输入你的选择：");  // 创建键盘录入对象  Scanner sc = **new** Scanner(System.***in***);  String choiceString = sc.nextLine();  // 用switch语句实现选择  **switch** (choiceString) {  **case** "1":  // 查看所有学生  *findAllStudent*(array);  **break**;  **case** "2":  // 添加学生  *addStudent*(array);  **break**;  **case** "3":  // 删除学生  *deleteStudent*(array);  **break**;  **case** "4":  // 修改学生  *updateStudent*(array);  **break**;  **case** "5":  // 退出  // System.out.println("谢谢你的使用");  // break;  **default**:  System.***out***.println("谢谢你的使用");  System.*exit*(0); // JVM退出  **break**;  }  }  }    /\*\*  \* **@Title**: findAllStudent  \* **@Description**: 查看所有学生  \* **@param** array  \*/  **public** **static** **void** findAllStudent(ArrayList<Student> array) {  //首先来判断集合中是否有数据，如果没有数据，就给出提示，并让该方法不继续往下执行  **if**(array.size() == 0) {  System.***out***.println("不好意思,目前没有学生信息可供查询,请回去重新选择你的操作");  **return**;  }    //\t 其实就是一个tab键的位置  System.***out***.println("学号\t姓名\t年龄\t居住地");  **for**(**int** x=0; x<array.size(); x++) {  Student s = array.get(x);  System.***out***.println(s.getId()+"\t"+s.getName()+"\t"+s.getAge()+"\t"+s.getAddress());  }  }    /\*  //添加学生  public static void addStudent(ArrayList<Student> array) {  //创建键盘录入对象  Scanner sc = new Scanner(System.in);    System.out.println("请输入学生学号：");  String id = sc.nextLine();  System.out.println("请输入学生姓名：");  String name = sc.nextLine();  System.out.println("请输入学生年龄：");  String age = sc.nextLine();  System.out.println("请输入学生居住地：");  String address = sc.nextLine();    //创建学生对象  Student s = new Student();  s.setId(id);  s.setName(name);  s.setAge(age);  s.setAddress(address);    //把学生对象作为元素添加到集合  array.add(s);    //给出提示  System.out.println("添加学生成功");  }  \*/    /\*\*  \* **@Title**: addStudent  \* **@Description**: 添加学生  \* **@param** array  \*/  **public** **static** **void** addStudent(ArrayList<Student> array) {  //创建键盘录入对象  Scanner sc = **new** Scanner(System.***in***);    //为了让id能够被访问到，我们就把id定义在了循环的外面  String id;    //为了让代码能够回到这里，用循环  **while**(**true**) {  System.***out***.println("请输入学生学号：");  //String id = sc.nextLine();  id = sc.nextLine();    //判断学号有没有被人占用  //定义标记  **boolean** flag = **false**;  //遍历集合，得到每一个学生  **for**(**int** x=0; x<array.size(); x++) {  Student s = array.get(x);  //获取该学生的学号，和键盘录入的学号进行比较  **if**(s.getId().equals(id)) {  flag = **true**; //说明学号被占用了  **break**;  }  }    **if**(flag) {  System.***out***.println("你输入的学号已经被占用,请重新输入");  }**else** {  **break**; //结束循环  }  }    System.***out***.println("请输入学生姓名：");  String name = sc.nextLine();  System.***out***.println("请输入学生年龄：");  String age = sc.nextLine();  System.***out***.println("请输入学生居住地：");  String address = sc.nextLine();    //创建学生对象  Student s = **new** Student();  s.setId(id);  s.setName(name);  s.setAge(age);  s.setAddress(address);    //把学生对象作为元素添加到集合  array.add(s);    //给出提示  System.***out***.println("添加学生成功");  }    /\*\*  \* **@Title**: deleteStudent  \* **@Description**: 删除学生  \* **@param** array  \*/  **public** **static** **void** deleteStudent(ArrayList<Student> array) {  //删除学生的思路：键盘录入一个学号，到集合中去查找，看是否有学生使用的是该学号，如果有就删除该学生  //创建键盘录入对象  Scanner sc = **new** Scanner(System.***in***);  System.***out***.println("请输入你要删除的学生的学号：");  String id = sc.nextLine();    /\*  //遍历集合  for(int x=0; x<array.size(); x++) {  //获取到每一个学生对象  Student s = array.get(x);  //拿这个学生对象的学号和键盘录入的学号进行比较  if(s.getId().equals(id)) {  array.remove(x); //根据索引删除  break;  }  }    //给出提示  System.out.println("删除学生成功");  \*/    //我们必须给出学号不存在的时候的提示    //定义一个索引  **int** index = -1;    //遍历集合  **for**(**int** x=0; x<array.size(); x++) {  //获取到每一个学生对象  Student s = array.get(x);  //拿这个学生对象的学号和键盘录入的学号进行比较  **if**(s.getId().equals(id)) {  index = x;  **break**;  }  }    **if**(index == -1) {  System.***out***.println("不好意思,你要删除的学号对应的学生信息不存在,请回去重新你的选择");  }**else** {  array.remove(index);  System.***out***.println("删除学生成功");  }    }    /\*\*  \* **@Title**: updateStudent  \* **@Description**: 修改学生  \* **@param** array  \*/  **public** **static** **void** updateStudent(ArrayList<Student> array) {  //修改学生的思路：键盘录入一个学号，到集合中去查找，看是否有学生使用的是该学号，如果有就修改该学生  //创建键盘录入对象  Scanner sc = **new** Scanner(System.***in***);  System.***out***.println("请输入你要修改的学生的学号：");  String id = sc.nextLine();    //定义一个索引  **int** index = -1;    //遍历集合  **for**(**int** x=0; x<array.size(); x++) {  //获取每一个学生对象  Student s = array.get(x);  //拿学生对象的学号和键盘录入的学号进行比较  **if**(s.getId().equals(id)) {  index = x;  **break**;  }  }    **if**(index == -1) {  System.***out***.println("不好意思,你要修改的学号对应的学生信息不存在,请回去重新你的选择");  }**else** {  System.***out***.println("请输入学生新姓名：");  String name = sc.nextLine();  System.***out***.println("请输入学生新年龄：");  String age = sc.nextLine();  System.***out***.println("请输入学生新居住地：");  String address = sc.nextLine();    //创建学生对象  Student s = **new** Student();  s.setId(id);  s.setName(name);  s.setAge(age);  s.setAddress(address);    //修改集合中的学生对象  array.set(index, s);    //给出提示  System.***out***.println("修改学生成功");  }  }  } |

# IO流概述及FileWriter类使用

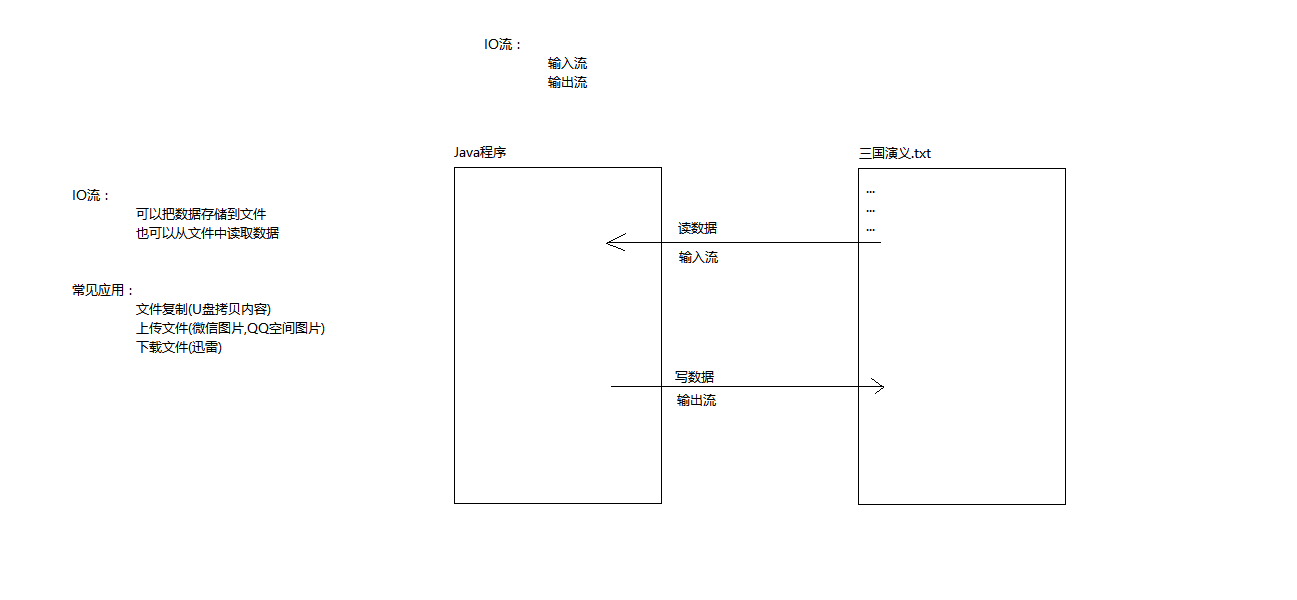
## IO流概述及分类

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

#### 案例代码十六

|  |
| --- |
| **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

#### 案例代码十七

|  |
| --- |
| **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

#### 案例代码十八

|  |
| --- |
| **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

#### 案例代码十九

|  |
| --- |
| **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

#### 案例代码二十

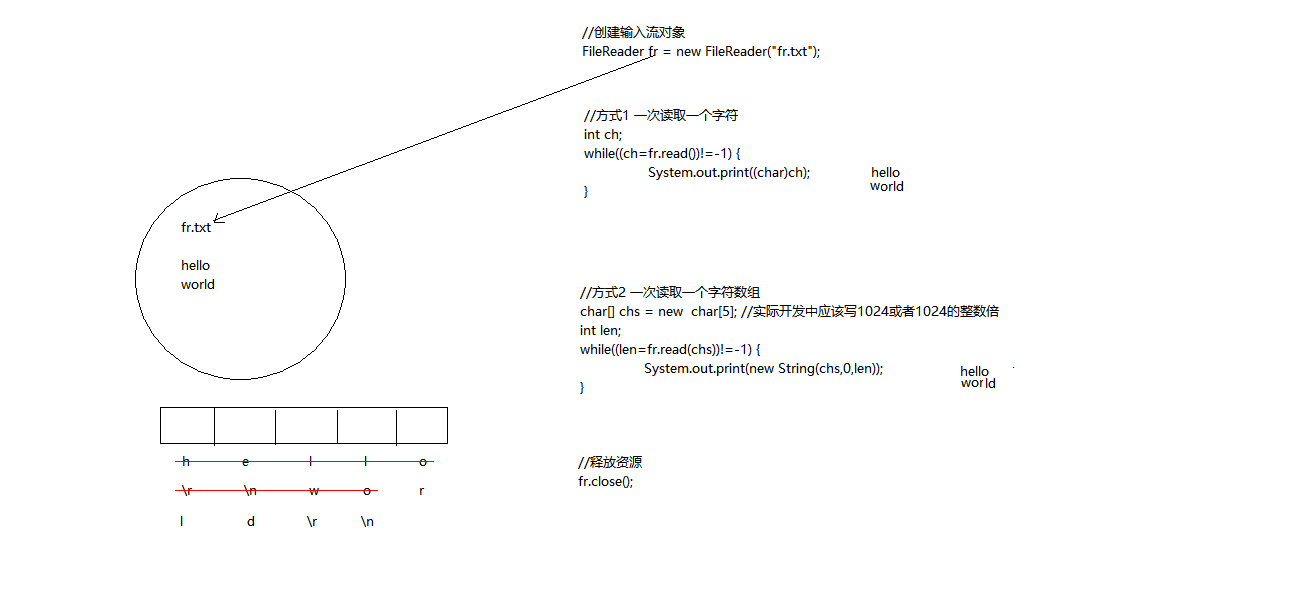
|  |
| --- |
| **package** com.igeek\_01;  **import** java.io.FileReader;  **import** java.io.IOException;  /\*\*  \* **@ClassName**: FileReaderDemo  \* **@Description**: 从文件中读数据并显示到控制台  \* **@date** 2017年10月14日 下午1:45:52  \* Company www.igeekhome.com  \*  \* 需求：从文件中读数据并显示到控制台  \* 读数据--输入流--FileReader  \*  \* 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

#### 案例代码二十一

|  |
| --- |
| **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

#### 案例代码二十二

|  |
| --- |
| **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

#### 案例代码二十三

|  |
| --- |
| **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

### 案例代码二十四

|  |
| --- |
| **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

### 案例代码二十五

|  |
| --- |
| **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

### 案例代码二十六

|  |
| --- |
| **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

### 案例代码二十七

|  |
| --- |
| **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

### 案例代码二十八

|  |
| --- |
| **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

### 案例代码二十九

|  |
| --- |
| **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

### 案例代码三十

|  |
| --- |
| **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);  }  }  } |

# 综合练习

## 综合练习一

A:键盘录入3个学生信息(学号,姓名,年龄,居住地)存入集合,要求学生信息的学号不能重复

B:遍历集合把每一个学生信息存入文本文件

C:每一个学生信息为一行数据,每行学生的学号,姓名,年龄和居住地在文件中均以逗号分隔

### 案例代码三十一

|  |
| --- |
| **package** com.igeek\_01;  /\*\*  \* **@ClassName**: Student  \* **@Description**: 学生类  \* **@date** 2017年10月14日 上午10:35:59  \* Company www.igeekhome.com  \*  \*/  **public** **class** Student {  /\*\*  \* **@Fields** id : 学号  \*/  **private** String id;  /\*\*  \* **@Fields** name : 姓名  \*/  **private** String name;  /\*\*  \* **@Fields** age : 年龄  \*/  **private** String age;  /\*\*  \* **@Fields** address : 居住地  \*/  **private** String address;  /\*\*  \* **@Title**: Student  \*/  **public** Student() {  }  /\*\*  \* **@Title**: Student  \* **@param** id  \* **@param** name  \* **@param** age  \* **@param** address  \*/  **public** Student(String id, String name, String age, String address) {  **this**.id = id;  **this**.name = name;  **this**.age = age;  **this**.address = address;  }  /\*\*  \* **@return** the id  \*/  **public** String getId() {  **return** id;  }  /\*\*  \* **@param** id the id to set  \*/  **public** **void** setId(String id) {  **this**.id = id;  }  /\*\*  \* **@return** the name  \*/  **public** String getName() {  **return** name;  }  /\*\*  \* **@param** name the name to set  \*/  **public** **void** setName(String name) {  **this**.name = name;  }  /\*\*  \* **@return** the age  \*/  **public** String getAge() {  **return** age;  }  /\*\*  \* **@param** age the age to set  \*/  **public** **void** setAge(String age) {  **this**.age = age;  }  /\*\*  \* **@return** the address  \*/  **public** String getAddress() {  **return** address;  }  /\*\*  \* **@param** address the address to set  \*/  **public** **void** setAddress(String address) {  **this**.address = address;  }  } |

|  |
| --- |
| **package** com.igeek\_01;  **import** java.io.BufferedWriter;  **import** java.io.FileWriter;  **import** java.io.IOException;  **import** java.util.ArrayList;  **import** java.util.Scanner;  /\*\*  \* **@ClassName**: ArrayListToFileTest  \* **@Description**: 学生信息添加到文件中  \* **@date** 2017年10月14日 下午2:57:18  \* Company www.igeekhome.com  \*  \* 键盘录入3个学生信息(学号,姓名,年龄,居住地)存入集合。然后遍历集合把每一个学生信息存入文本文件(每一个学生信息为一行数据，自己定义分割标记)  \*  \* 分析：  \* A:定义学生类  \* B:创建集合对象  \* C:写方法实现键盘录入学生信息，并把学生对象作为元素添加到集合  \* D:创建输出缓冲流对象  \* E:遍历集合,得到每一个学生信息，并把学生信息按照一定的格式写入文本文件  \* 举例：1001,jim,18,无锡  \* F:释放资源  \*/  **public** **class** ArrayListToFileTest {  **public** **static** **void** main(String[] args) **throws** IOException {  //创建集合对象  ArrayList<Student> array = **new** ArrayList<Student>();    //写方法实现键盘录入学生信息，并把学生对象作为元素添加到集合  *addStudent*(array);  *addStudent*(array);  *addStudent*(array);    //创建输出缓冲流对象  BufferedWriter bw = **new** BufferedWriter(**new** FileWriter("array.txt"));    //遍历集合,得到每一个学生信息，并把学生信息按照一定的格式写入文本文件  **for**(**int** x=0; x<array.size(); x++) {  Student s = array.get(x);  //1001,jim,18,无锡  StringBuilder sb = **new** StringBuilder();  sb.append(s.getId()).append(",").append(s.getName()).append(",").append(s.getAge()).append(",").append(s.getAddress());    bw.write(sb.toString());  bw.newLine();  bw.flush();  }    //释放资源  bw.close();  }    /\*\*  \* **@Title**: addStudent  \* **@Description**: 添加学生  \* **@param** array  \*/  **public** **static** **void** addStudent(ArrayList<Student> array) {  //创建键盘录入对象  Scanner sc = **new** Scanner(System.***in***);    //为了让id能够被访问到，我们就把id定义在了循环的外面  String id;    //为了让代码能够回到这里，用循环  **while**(**true**) {  System.***out***.println("请输入学生学号：");  //String id = sc.nextLine();  id = sc.nextLine();    //判断学号有没有被人占用  //定义标记  **boolean** flag = **false**;  //遍历集合，得到每一个学生  **for**(**int** x=0; x<array.size(); x++) {  Student s = array.get(x);  //获取该学生的学号，和键盘录入的学号进行比较  **if**(s.getId().equals(id)) {  flag = **true**; //说明学号被占用了  **break**;  }  }    **if**(flag) {  System.***out***.println("你输入的学号已经被占用,请重新输入");  }**else** {  **break**; //结束循环  }  }      System.***out***.println("请输入学生姓名：");  String name = sc.nextLine();  System.***out***.println("请输入学生年龄：");  String age = sc.nextLine();  System.***out***.println("请输入学生居住地：");  String address = sc.nextLine();    //创建学生对象  Student s = **new** Student();  s.setId(id);  s.setName(name);  s.setAge(age);  s.setAddress(address);    //把学生对象作为元素添加到集合  array.add(s);    //给出提示  System.***out***.println("添加学生成功");  }  } |

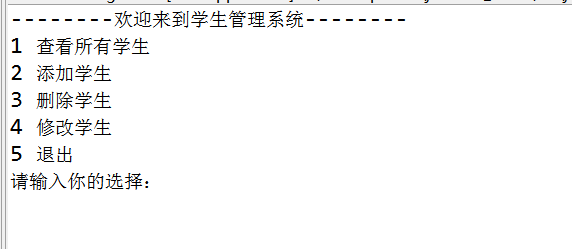
## 综合练习二

把上一案例的array.txt文本文件中的学生信息读取出来存储到集合中，然后遍历集合，在控制台输出

### 案例代码三十二

|  |
| --- |
| **package** com.igeek\_02;  **import** java.io.BufferedReader;  **import** java.io.FileReader;  **import** java.io.IOException;  **import** java.util.ArrayList;  /\*\*  \* **@ClassName**: FileToArrayListTest  \* **@Description**: 把文本文件中的学生信息读取出来存储到集合中  \* **@date** 2017年10月14日 下午3:04:38  \* Company www.igeekhome.com  \*  \* 把上一题的文本文件中的学生信息读取出来存储到集合中，然后遍历集合，在控制台输出  \*  \* 分析：  \* A:定义学生类  \* B:创建输入缓冲流对象  \* C:创建集合对象  \* D:读取文件数据，并把数据按照一定的格式进行分割赋值给学生对象，然后把学生对象作为元素存储到集合  \* 1001,jim,18,无锡  \* E:释放资源  \* F: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<Student> array = **new** ArrayList<Student>();    //读取文件数据，并把数据按照一定的格式进行分割赋值给学生对象，然后把学生对象作为元素存储到集合  String line;  **while**((line=br.readLine())!=**null**) {  //分割字符串  String[] strArray = line.split(",");  //创建学生对象并赋值  Student s = **new** Student();  s.setId(strArray[0]);  s.setName(strArray[1]);  s.setAge(strArray[2]);  s.setAddress(strArray[3]);  //把学生对象作为元素存储到集合  array.add(s);  }    //释放资源  br.close();    //遍历集合  System.***out***.println("学号\t姓名\t年龄\t居住地");  **for**(**int** x=0; x<array.size(); x++) {  Student s = array.get(x);  System.***out***.println(s.getId()+"\t"+s.getName()+"\t"+s.getAge()+"\t"+s.getAddress());  }  }  } |

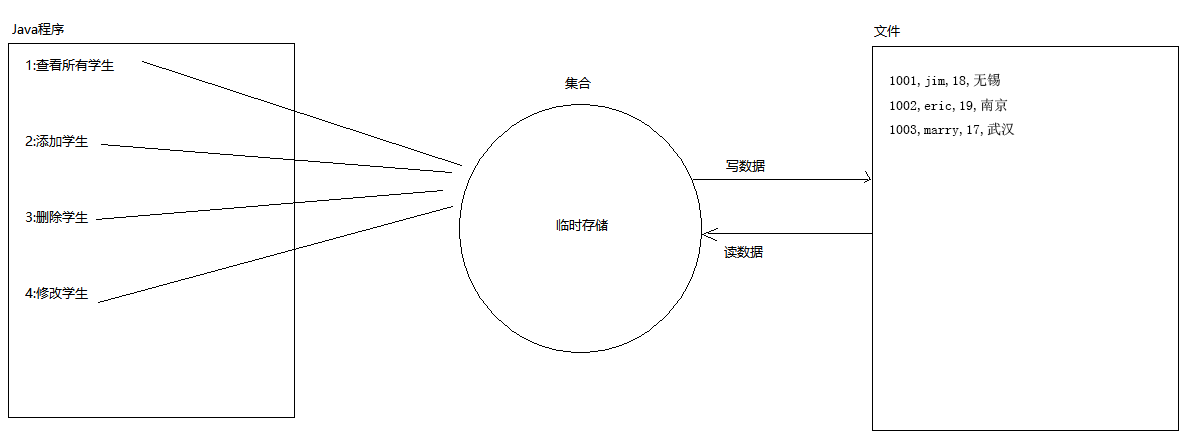
## 综合案例三



学生管理系统IO版:

A:在第五天上午学生管理系统案例的基础上,添加新需求

B:查看所有的学生的数据需要从students.txt(提供好的)中读取出来

 C:增删改都需要先从students.txt中读出学生的数据然后再把改后的学生数据重新写回students.txt

### 原有的集合版学生管理系统

#### 案例代码三十三

|  |
| --- |
| **package** com.igeek\_03;  **import** java.util.ArrayList;  **import** java.util.Scanner;  /\*\*  \* **@ClassName**: StudentManagerTest  \* **@Description**: 学生管理系统的主类  \* **@date** 2017年10月14日 上午10:50:19  \* Company www.igeekhome.com  \*  \* 这是我的学生管理系统的主类  \*  \* 步骤如下：  \* A:定义学生类  \* B:学生管理系统的主界面的代码编写  \* C:学生管理系统的查看所有学生的代码编写  \* D:学生管理系统的添加学生的代码编写  \* E:学生管理系统的删除学生的代码编写  \* F:学生管理系统的修改学生的代码编写  \*/  **public** **class** StudentManagerTest {  **public** **static** **void** main(String[] args) {  // 创建集合对象，用于存储学生数据  ArrayList<Student> array = **new** ArrayList<Student>();  // 为了让程序能够回到这里来，我们使用循环  **while** (**true**) {  // 这是学生管理系统的主界面  System.***out***.println("--------欢迎来到学生管理系统--------");  System.***out***.println("1 查看所有学生");  System.***out***.println("2 添加学生");  System.***out***.println("3 删除学生");  System.***out***.println("4 修改学生");  System.***out***.println("5 退出");  System.***out***.println("请输入你的选择：");  // 创建键盘录入对象  Scanner sc = **new** Scanner(System.***in***);  String choiceString = sc.nextLine();  // 用switch语句实现选择  **switch** (choiceString) {  **case** "1":  // 查看所有学生  *findAllStudent*(array);  **break**;  **case** "2":  // 添加学生  *addStudent*(array);  **break**;  **case** "3":  // 删除学生  *deleteStudent*(array);  **break**;  **case** "4":  // 修改学生  *updateStudent*(array);  **break**;  **case** "5":  // 退出  // System.out.println("谢谢你的使用");  // break;  **default**:  System.***out***.println("谢谢你的使用");  System.*exit*(0); // JVM退出  **break**;  }  }  }    /\*\*  \* **@Title**: findAllStudent  \* **@Description**: 查看所有学生  \* **@param** array  \*/  **public** **static** **void** findAllStudent(ArrayList<Student> array) {  //首先来判断集合中是否有数据，如果没有数据，就给出提示，并让该方法不继续往下执行  **if**(array.size() == 0) {  System.***out***.println("不好意思,目前没有学生信息可供查询,请回去重新选择你的操作");  **return**;  }    //\t 其实就是一个tab键的位置  System.***out***.println("学号\t姓名\t年龄\t居住地");  **for**(**int** x=0; x<array.size(); x++) {  Student s = array.get(x);  System.***out***.println(s.getId()+"\t"+s.getName()+"\t"+s.getAge()+"\t"+s.getAddress());  }  }    /\*\*  \* **@Title**: addStudent  \* **@Description**: 添加学生  \* **@param** array  \*/  **public** **static** **void** addStudent(ArrayList<Student> array) {  //创建键盘录入对象  Scanner sc = **new** Scanner(System.***in***);    //为了让id能够被访问到，我们就把id定义在了循环的外面  String id;    //为了让代码能够回到这里，用循环  **while**(**true**) {  System.***out***.println("请输入学生学号：");  //String id = sc.nextLine();  id = sc.nextLine();    //判断学号有没有被人占用  //定义标记  **boolean** flag = **false**;  //遍历集合，得到每一个学生  **for**(**int** x=0; x<array.size(); x++) {  Student s = array.get(x);  //获取该学生的学号，和键盘录入的学号进行比较  **if**(s.getId().equals(id)) {  flag = **true**; //说明学号被占用了  **break**;  }  }    **if**(flag) {  System.***out***.println("你输入的学号已经被占用,请重新输入");  }**else** {  **break**; //结束循环  }  }    System.***out***.println("请输入学生姓名：");  String name = sc.nextLine();  System.***out***.println("请输入学生年龄：");  String age = sc.nextLine();  System.***out***.println("请输入学生居住地：");  String address = sc.nextLine();    //创建学生对象  Student s = **new** Student();  s.setId(id);  s.setName(name);  s.setAge(age);  s.setAddress(address);    //把学生对象作为元素添加到集合  array.add(s);    //给出提示  System.***out***.println("添加学生成功");  }    /\*\*  \* **@Title**: deleteStudent  \* **@Description**: 删除学生  \* **@param** array  \*/  **public** **static** **void** deleteStudent(ArrayList<Student> array) {  //删除学生的思路：键盘录入一个学号，到集合中去查找，看是否有学生使用的是该学号，如果有就删除该学生  //创建键盘录入对象  Scanner sc = **new** Scanner(System.***in***);  System.***out***.println("请输入你要删除的学生的学号：");  String id = sc.nextLine();    //我们必须给出学号不存在的时候的提示    //定义一个索引  **int** index = -1;    //遍历集合  **for**(**int** x=0; x<array.size(); x++) {  //获取到每一个学生对象  Student s = array.get(x);  //拿这个学生对象的学号和键盘录入的学号进行比较  **if**(s.getId().equals(id)) {  index = x;  **break**;  }  }    **if**(index == -1) {  System.***out***.println("不好意思,你要删除的学号对应的学生信息不存在,请回去重新你的选择");  }**else** {  array.remove(index);  System.***out***.println("删除学生成功");  }    }    /\*\*  \* **@Title**: updateStudent  \* **@Description**: 修改学生  \* **@param** array  \*/  **public** **static** **void** updateStudent(ArrayList<Student> array) {  //修改学生的思路：键盘录入一个学号，到集合中去查找，看是否有学生使用的是该学号，如果有就修改该学生  //创建键盘录入对象  Scanner sc = **new** Scanner(System.***in***);  System.***out***.println("请输入你要修改的学生的学号：");  String id = sc.nextLine();    //定义一个索引  **int** index = -1;    //遍历集合  **for**(**int** x=0; x<array.size(); x++) {  //获取每一个学生对象  Student s = array.get(x);  //拿学生对象的学号和键盘录入的学号进行比较  **if**(s.getId().equals(id)) {  index = x;  **break**;  }  }    **if**(index == -1) {  System.***out***.println("不好意思,你要修改的学号对应的学生信息不存在,请回去重新你的选择");  }**else** {  System.***out***.println("请输入学生新姓名：");  String name = sc.nextLine();  System.***out***.println("请输入学生新年龄：");  String age = sc.nextLine();  System.***out***.println("请输入学生新居住地：");  String address = sc.nextLine();    //创建学生对象  Student s = **new** Student();  s.setId(id);  s.setName(name);  s.setAge(age);  s.setAddress(address);    //修改集合中的学生对象  array.set(index, s);    //给出提示  System.***out***.println("修改学生成功");  }  }  } |

### 对文本文件进行读写操作

#### 案例代码三十四

public static void readData(String fileName,ArrayList<Student> array):从fileName的文件中读取学生的数据,并把学生的数据封装到array集合中

public static void writeData(String fileName,ArrayList<Student> array):

将array中的每个元素中包含的数据写入到fileName的文件中

|  |
| --- |
| /\*\*  \* **@Title**: readData  \* **@Description**: 从文件中读数据到集合  \* **@param** fileName  \* **@param** array  \* **@throws** IOException  \*/  **public** **static** **void** readData(String fileName, ArrayList<Student> array)  **throws** IOException {  // 创建输入缓冲流对象  BufferedReader br = **new** BufferedReader(**new** FileReader(fileName));  String line;  **while** ((line = br.readLine()) != **null**) {  String[] datas = line.split(",");  Student s = **new** Student();  s.setId(datas[0]);  s.setName(datas[1]);  s.setAge(datas[2]);  s.setAddress(datas[3]);  array.add(s);  }  br.close();  }  /\*\*  \* **@Title**: writeData  \* **@Description**: 把集合中的数据写入文件  \* **@param** fileName  \* **@param** array  \* **@throws** IOException  \*/  **public** **static** **void** writeData(String fileName, ArrayList<Student> array)  **throws** IOException {  // 创建输出缓冲流对象  BufferedWriter bw = **new** BufferedWriter(**new** FileWriter(fileName));  **for** (**int** x = 0; x < array.size(); x++) {  Student s = array.get(x);  StringBuilder sb = **new** StringBuilder();  sb.append(s.getId()).append(",").append(s.getName()).append(",")  .append(s.getAge()).append(",").append(s.getAddress());  bw.write(sb.toString());  bw.newLine();  bw.flush();  }  bw.close();  } |

### 增删改查功能调用读写文件方法

#### 案例代码三十五

|  |
| --- |
| **package** com.igeek\_03\_03;  **import** java.io.BufferedReader;  **import** java.io.BufferedWriter;  **import** java.io.FileReader;  **import** java.io.FileWriter;  **import** java.io.IOException;  **import** java.util.ArrayList;  **import** java.util.Scanner;  /\*\*  \* **@ClassName**: StudentManagerTest  \* **@Description**: 学生管理系统的主类  \* **@date** 2017年10月14日 上午10:50:19  \* Company www.igeekhome.com  \*  \* 这是我的学生管理系统的主类  \*  \* 步骤如下：  \* A:定义学生类  \* B:学生管理系统的主界面的代码编写  \* C:学生管理系统的查看所有学生的代码编写  \* D:学生管理系统的添加学生的代码编写  \* E:学生管理系统的删除学生的代码编写  \* F:学生管理系统的修改学生的代码编写  \*/  **public** **class** StudentManagerTest {  **public** **static** **void** main(String[] args) **throws** IOException {  //定义文件路径  String fileName = "students.txt";    //为了让程序能够回到这里来，我们使用循环  **while**(**true**) {  //这是学生管理系统的主界面  System.***out***.println("--------欢迎来到学生管理系统--------");  System.***out***.println("1 查看所有学生");  System.***out***.println("2 添加学生");  System.***out***.println("3 删除学生");  System.***out***.println("4 修改学生");  System.***out***.println("5 退出");  System.***out***.println("请输入你的选择：");  //创建键盘录入对象  Scanner sc = **new** Scanner(System.***in***);  String choiceString = sc.nextLine();  //用switch语句实现选择  **switch**(choiceString) {  **case** "1":  //查看所有学生  *findAllStudent*(fileName);  **break**;  **case** "2":  //添加学生  *addStudent*(fileName);  **break**;  **case** "3":  //删除学生  *deleteStudent*(fileName);  **break**;  **case** "4":  //修改学生  *updateStudent*(fileName);  **break**;  **case** "5":  **default**:  System.***out***.println("谢谢你的使用");  System.*exit*(0); //JVM退出  **break**;  }  }  }    /\*\*  \* **@Title**: readData  \* **@Description**: 从文件中读数据到集合  \* **@param** fileName  \* **@param** array  \* **@throws** IOException  \*/  **public** **static** **void** readData(String fileName, ArrayList<Student> array)  **throws** IOException {  // 创建输入缓冲流对象  BufferedReader br = **new** BufferedReader(**new** FileReader(fileName));  String line;  **while** ((line = br.readLine()) != **null**) {  String[] datas = line.split(",");  Student s = **new** Student();  s.setId(datas[0]);  s.setName(datas[1]);  s.setAge(datas[2]);  s.setAddress(datas[3]);  array.add(s);  }  br.close();  }  /\*\*  \* **@Title**: writeData  \* **@Description**: 把集合中的数据写入文件  \* **@param** fileName  \* **@param** array  \* **@throws** IOException  \*/  **public** **static** **void** writeData(String fileName, ArrayList<Student> array)  **throws** IOException {  // 创建输出缓冲流对象  BufferedWriter bw = **new** BufferedWriter(**new** FileWriter(fileName));  **for** (**int** x = 0; x < array.size(); x++) {  Student s = array.get(x);  StringBuilder sb = **new** StringBuilder();  sb.append(s.getId()).append(",").append(s.getName()).append(",")  .append(s.getAge()).append(",").append(s.getAddress());  bw.write(sb.toString());  bw.newLine();  bw.flush();  }  bw.close();  }    /\*\*  \* **@Title**: updateStudent  \* **@Description**: 修改学生  \* **@param** fileName  \* **@throws** IOException  \*/  **public** **static** **void** updateStudent(String fileName) **throws** IOException {  //创建集合对象  ArrayList<Student> array = **new** ArrayList<Student>();  //从文件中把数据读取到集合中  *readData*(fileName, array);    //修改学生的思路：键盘录入一个学号，到集合中去查找，看是否有学生使用的是该学号，如果有就修改该学生  //创建键盘录入对象  Scanner sc = **new** Scanner(System.***in***);  System.***out***.println("请输入你要修改的学生的学号：");  String id = sc.nextLine();    //定义一个索引  **int** index = -1;    //遍历集合  **for**(**int** x=0; x<array.size(); x++) {  //获取每一个学生对象  Student s = array.get(x);  //拿学生对象的学号和键盘录入的学号进行比较  **if**(s.getId().equals(id)) {  index = x;  **break**;  }  }    **if**(index == -1) {  System.***out***.println("不好意思,你要修改的学号对应的学生信息不存在,请回去重新你的选择");  }**else** {  System.***out***.println("请输入学生新姓名：");  String name = sc.nextLine();  System.***out***.println("请输入学生新年龄：");  String age = sc.nextLine();  System.***out***.println("请输入学生新居住地：");  String address = sc.nextLine();    //创建学生对象  Student s = **new** Student();  s.setId(id);  s.setName(name);  s.setAge(age);  s.setAddress(address);    //修改集合中的学生对象  array.set(index, s);  //把集合中的数据重新写回到文件  *writeData*(fileName, array);  //给出提示  System.***out***.println("修改学生成功");  }  }    /\*\*  \* **@Title**: deleteStudent  \* **@Description**: 删除学生  \* **@param** fileName  \* **@throws** IOException  \*/  **public** **static** **void** deleteStudent(String fileName) **throws** IOException {  //创建集合对象  ArrayList<Student> array = **new** ArrayList<Student>();  //从文件中把数据读取到集合中  *readData*(fileName, array);    //删除学生的思路：键盘录入一个学号，到集合中去查找，看是否有学生使用的是该学号，如果有就删除该学生  //创建键盘录入对象  Scanner sc = **new** Scanner(System.***in***);  System.***out***.println("请输入你要删除的学生的学号：");  String id = sc.nextLine();    //我们必须给出学号不存在的时候的提示    //定义一个索引  **int** index = -1;    //遍历集合  **for**(**int** x=0; x<array.size(); x++) {  //获取到每一个学生对象  Student s = array.get(x);  //拿这个学生对象的学号和键盘录入的学号进行比较  **if**(s.getId().equals(id)) {  index = x;  **break**;  }  }    **if**(index == -1) {  System.***out***.println("不好意思,你要删除的学号对应的学生信息不存在,请回去重新你的选择");  }**else** {  array.remove(index);  //把集合中的数据重新写回到文件  *writeData*(fileName, array);  System.***out***.println("删除学生成功");  }    }    /\*\*  \* **@Title**: addStudent  \* **@Description**: 添加学生  \* **@param** fileName  \* **@throws** IOException  \*/  **public** **static** **void** addStudent(String fileName) **throws** IOException {  //创建集合对象  ArrayList<Student> array = **new** ArrayList<Student>();  //从文件中把数据读取到集合中  *readData*(fileName, array);    //创建键盘录入对象  Scanner sc = **new** Scanner(System.***in***);    //为了让id能够被访问到，我们就把id定义在了循环的外面  String id;    //为了让代码能够回到这里，用循环  **while**(**true**) {  System.***out***.println("请输入学生学号：");  //String id = sc.nextLine();  id = sc.nextLine();    //判断学号有没有被人占用  //定义标记  **boolean** flag = **false**;  //遍历集合，得到每一个学生  **for**(**int** x=0; x<array.size(); x++) {  Student s = array.get(x);  //获取该学生的学号，和键盘录入的学号进行比较  **if**(s.getId().equals(id)) {  flag = **true**; //说明学号被占用了  **break**;  }  }    **if**(flag) {  System.***out***.println("你输入的学号已经被占用,请重新输入");  }**else** {  **break**; //结束循环  }  }    System.***out***.println("请输入学生姓名：");  String name = sc.nextLine();  System.***out***.println("请输入学生年龄：");  String age = sc.nextLine();  System.***out***.println("请输入学生居住地：");  String address = sc.nextLine();    //创建学生对象  Student s = **new** Student();  s.setId(id);  s.setName(name);  s.setAge(age);  s.setAddress(address);    //把学生对象作为元素添加到集合  array.add(s);  //把集合中的数据重新写回到文件  *writeData*(fileName, array);    //给出提示  System.***out***.println("添加学生成功");  }    /\*\*  \* **@Title**: findAllStudent  \* **@Description**: 查看所有学生  \* **@param** fileName  \* **@throws** IOException  \*/  **public** **static** **void** findAllStudent(String fileName) **throws** IOException {  //创建集合对象  ArrayList<Student> array = **new** ArrayList<Student>();  //从文件中把数据读取到集合中  *readData*(fileName, array);    //首先来判断集合中是否有数据，如果没有数据，就给出提示，并让该方法不继续往下执行  **if**(array.size() == 0) {  System.***out***.println("不好意思,目前没有学生信息可供查询,请回去重新选择你的操作");  **return**;  }    //\t 其实就是一个tab键的位置  System.***out***.println("学号\t姓名\t年龄\t居住地");  **for**(**int** x=0; x<array.size(); x++) {  Student s = array.get(x);  System.***out***.println(s.getId()+"\t"+s.getName()+"\t"+s.getAge()+"\t"+s.getAddress());  }  }  } |

重点和总结

1、对象数组

2、集合类之ArrayList

4、IO流概述及FileWriter类使用

5、FileReader类使用

6、缓冲流介绍和使用