Java进阶1 第13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】面向对象练习**
2. 【应用】能够独立分析类与类,类与接口的继承或实现关系
3. 【应用】能够根据案例定义出抽象类和接口,并测试
4. **【应用】API练习**
5. 【应用】能够独立写出删除一个字符串中指定的子串案例
6. **【应用】集合练习**
7. 【应用】 能够独立完成统计每个班级的平均分和总分的案例
8. **【应用】IO递归练习**
9. 【应用】 能够独立写出文件夹递归拷贝案例

# 面向对象练习

## 需求:

描述笔记本类，主要实现笔记本使用USB设备功能，最终达到要求的打印结果。

笔记本类:

开机功能

关机功能

定义使用USB设备的功能，要求：既能使用鼠标也能使用键盘，使用 USB功能内部调用开启和关闭功能

USB接口:

开启功能

工作方法

关闭功能

硬件设备类:

工作方法

电脑屏幕类:

工作方法为显示画面

鼠标类:

工作方法为点击

属于硬件设备，但要符合USB接口

键盘类:

工作方法为按键

属于硬件设备，但要符合USB接口

测试类:

创建电脑对象，依次调用开机方法，使用USB设备，关机方法

打印效果如下:

开机

显示屏显示画面

连接鼠标的USB

鼠标点击

断开鼠标的USB

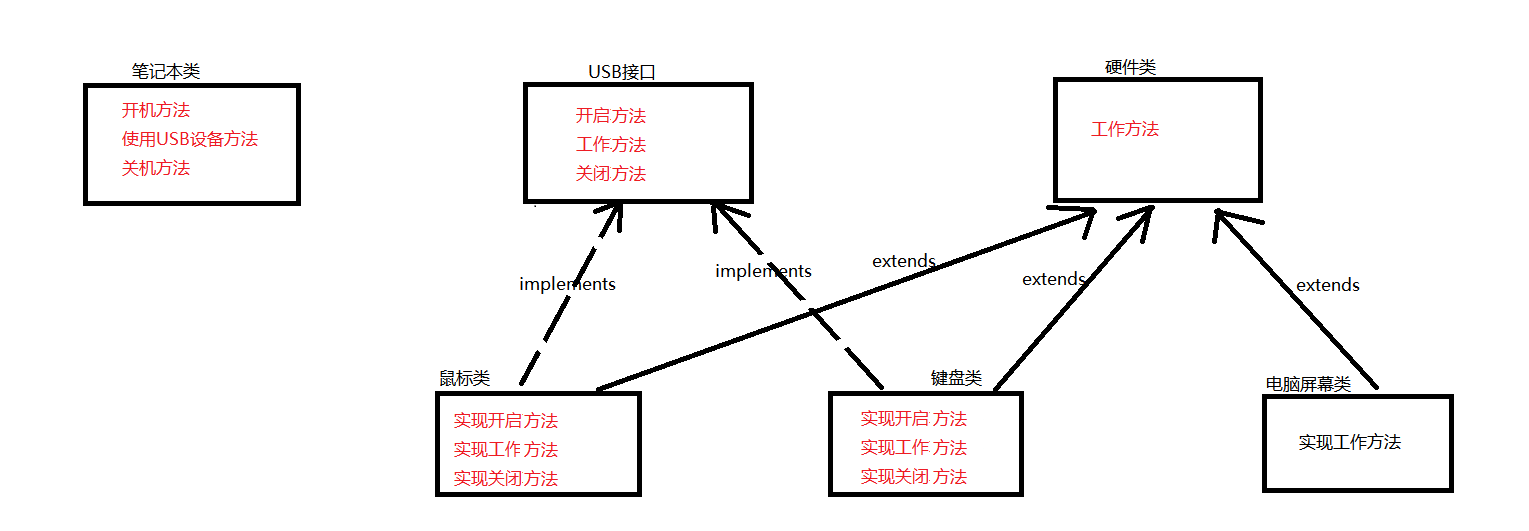
连接键盘的USB

键盘按键

断开键盘的USB

关机

## 需求分析:



## 需求实现

### 案例代码一:

|  |
| --- |
| **package** com.igeek\_01;  /\*\*  \* **@ClassName**: USB  \* **@Description**: USB接口  \* **@date** 2017年11月24日 上午9:51:43  \* Company www.igeekhome.com  \*  \* USB接口中方法:  \* 开启功能参数:无 返回值:无  \* 工作方法参数:无 返回值:无  \* 关闭功能参数:无 返回值:无  \*/  **public** **interface** USB {    /\*\*  \* **@Title**: open  \* **@Description**: 开启功能  \*/  **public** **abstract** **void** open();    /\*\*  \* **@Title**: work  \* **@Description**: 工作功能  \*/  **public** **abstract** **void** work();    /\*\*  \* **@Title**: close  \* **@Description**: 关闭功能  \*/  **public** **abstract** **void** close();  } |

|  |
| --- |
| **package** com.igeek\_01;  /\*\*  \* **@ClassName**: Hardware  \* **@Description**: 硬件设备类  \* **@date** 2017年11月24日 上午9:53:40  \* Company www.igeekhome.com  \*  \* 硬件设备类方法:  \* 工作方法参数:无  \* 返回值:无  \* 方法内容:由于硬件设备可以设计为抽象类,所以工作方法设计为抽象方法,无方法体  \*/  **public** **abstract** **class** Hardware {  /\*\*  \* **@Title**: work  \* **@Description**: 工作方法  \*/  **public** **abstract** **void** work();  } |

|  |
| --- |
| **package** com.igeek\_01;  /\*\*  \* **@ClassName**: Mouse  \* **@Description**: 鼠标类  \* **@date** 2017年11月24日 上午9:57:23  \* Company www.igeekhome.com  \*  \* 鼠标是硬件类的子类,是USB接口的实现类  \*  \* 重写方法:  \* 开启功能参数:无 返回值:无 方法内容:连接鼠标  \* 工作方法参数:无 返回值:无 方法内容:打印鼠标点击  \* 关闭功能参数:无 返回值:无 方法内容:断开鼠标  \*/  **public** **class** Mouse **extends** Hardware **implements** USB {  /\*\*  \* **@Title**: open  \* **@Description**: 开启功能  \* **@see** com.igeek\_01.USB#open()  \*/  @Override  **public** **void** open() {  System.***out***.println("连接鼠标");  }  /\*\*  \* **@Title**: close  \* **@Description**: 关闭功能  \* **@see** com.igeek\_01.USB#close()  \*/  @Override  **public** **void** close() {  System.***out***.println("断开鼠标");  }  /\*\*  \* **@Title**: work  \* **@Description**: 工作方法  \* **@see** com.igeek\_01.Hardware#work()  \*/  @Override  **public** **void** work() {  System.***out***.println("鼠标点击");  }  } |

|  |
| --- |
| **package** com.igeek\_01;  /\*\*  \* **@ClassName**: Keyboard  \* **@Description**: 键盘类  \* **@date** 2017年11月24日 上午10:00:30  \* Company www.igeekhome.com  \*  \* 键盘是硬件类的子类,是USB接口的实现类  \*  \* 重写方法:  \* 开启功能参数:无 返回值:无 方法内容:连接键盘  \* 工作方法参数:无 返回值:无 方法内容:打印键盘按键  \* 关闭功能参数:无 返回值:无 方法内容:断开键盘  \*/  **public** **class** Keyboard **extends** Hardware **implements** USB {  /\*\*  \* **@Title**: open  \* **@Description**: 开启功能  \* **@see** com.igeek\_01.USB#open()  \*/  @Override  **public** **void** open() {  System.***out***.println("连接键盘");  }  /\*\*  \* **@Title**: close  \* **@Description**: 关闭功能  \* **@see** com.igeek\_01.USB#close()  \*/  @Override  **public** **void** close() {  System.***out***.println("断开键盘");  }  /\*\*  \* **@Title**: work  \* **@Description**: 工作方法  \* **@see** com.igeek\_01.Hardware#work()  \*/  @Override  **public** **void** work() {  System.***out***.println("键盘按键");  }  } |

|  |
| --- |
| **package** com.igeek\_01;  /\*\*  \* **@ClassName**: Screen  \* **@Description**: 电脑屏幕子类  \* **@date** 2017年11月24日 上午10:04:42  \* Company www.igeekhome.com  \*  \* 电脑屏幕是硬件类的子类  \*  \* 重写方法:  \* 工作方法参数:无 返回值:无 方法内容:打印显示屏显示画面  \*/  **public** **class** Screen **extends** Hardware {  /\*\*  \* **@Title**: work  \* **@Description**: 工作方法  \* **@see** com.igeek\_01.Hardware#work()  \*/  @Override  **public** **void** work() {  System.***out***.println("显示屏显示画面 ");  }  } |

|  |
| --- |
| **package** com.igeek\_01;  /\*\*  \* **@ClassName**: Computer  \* **@Description**: 笔记本类  \* **@date** 2017年11月24日 上午10:07:19  \* Company www.igeekhome.com  \*  \* 笔记本类中方法:  \* 开机功能参数:无 返回值:无 功能内容:打印开机,创建电脑屏幕类对象,调用显示画面方法  \* 关机功能参数:无 返回值:无 功能内容:打印关机  \* 使用USB设备的功能  \* 参数:USB接口参数(形式参数)  \* 返回值:无  \* 功能内容:调用USB设备的三个功能:开启功能,工作方法,关闭功能  \*/  **public** **class** Computer {    /\*\*  \* **@Title**: begin  \* **@Description**: 开机方法  \*/  **public** **void** begin() {  System.***out***.println("开机");  //创建电脑屏幕类对象  Screen sc = **new** Screen();  //调用显示画面方法  sc.work();  }    /\*\*  \* **@Title**: over  \* **@Description**: 关机方法  \*/  **public** **void** over() {  System.***out***.println("关机");  }    /\*\*  \* **@Title**: useUSB  \* **@Description**: 使用USB功能  \* **@param** usb  \*/  **public** **void** useUSB(USB usb) {  usb.open();  usb.work();  usb.close();  }  } |

|  |
| --- |
| **package** com.igeek\_01;  /\*\*  \* **@ClassName**: ComputerTest  \* **@Description**: 面向对象练习题,计算机综合测试类  \* **@date** 2017年11月24日 上午10:14:38  \* Company www.igeekhome.com  \*  面向对象题目:    分析以下需求，并用代码实现:  描述笔记本类，主要实现笔记本使用USB设备功能，最终达到要求的打印结果。  笔记本类:  开机功能  关机功能  定义使用USB设备的功能，要求：既能使用鼠标也能使用键盘，使用USB功能内部调用开启和关闭功能    USB接口:  开启功能  工作方法  关闭功能    硬件设备类:  工作方法    电脑屏幕类:  工作方法为显示画面    鼠标类:  工作方法为点击  属于硬件设备，但要符合USB接口    键盘类:  工作方法为按键  属于硬件设备，但要符合USB接口    测试类:  创建电脑对象，依次调用开机方法，使用USB设备，关机方法    打印效果如下:  开机  显示屏显示画面  连接鼠标的USB  鼠标点击  断开鼠标的USB  连接键盘的USB  键盘按键  断开键盘的USB  关机  ========================================  \* 分析过程一:类间关系  \*  \* 笔记本类使用其他的类  \* USB接口  \* 鼠标  \* 键盘  \* 硬件设备类  \* 电脑屏幕类  \* 鼠标  \* 键盘  \* 测试类创建笔记本类对象调用方法  \*  \* 分析过程二:类中功能(方法)  \* 笔记本类中方法:  \* 开机功能 参数:无 返回值:无 功能内容:打印开机,创建电脑屏幕类对象,调用显示画面方法  \* 关机功能 参数:无 返回值:无 功能内容:打印关机  \* 使用USB设备的功能  \* 参数:USB接口参数(形式参数)  \* 返回值:无  \* 功能内容:调用USB设备的三个功能:开启功能,工作方法,关闭功能  \*  \* USB接口中方法:  \* 开启功能 参数:无 返回值:无  \* 工作方法 参数:无 返回值:无  \* 关闭功能 参数:无 返回值:无  \*  \* 硬件设备类方法:  \* 工作方法 参数:无 返回值:无 方法内容:由于硬件设备可以设计为抽象类,所以工作方法设计为抽象方法,无方法体  \*  \* 电脑屏幕类:  \* 工作方法 参数:无 返回值:无 方法内容:打印显示屏显示画面  \*  \* 鼠标类:  \* 开启功能 参数:无 返回值:无 方法内容:连接鼠标  \* 工作方法 参数:无 返回值:无 方法内容:打印鼠标点击  \* 关闭功能 参数:无 返回值:无 方法内容:断开鼠标  \*  \* 键盘类:  \* 开启功能 参数:无 返回值:无 方法内容:连接键盘  \* 工作方法 参数:无 返回值:无 方法内容:打印键盘按键  \* 关闭功能 参数:无 返回值:无 方法内容:断开键盘  \*  \* 分析过程三:完成测试  \* 测试类创建笔记本对象,调用笔记本对象的方法:  \* 开机方法  \* 使用USB设备方法:  \* 创建鼠标对象  \* 调用使用USB设备方法  \* 使用USB设备方法:  \* 创建键盘对象  \* 调用使用USB设备方法  \* 关机方法  \*/  **public** **class** ComputerTest {  **public** **static** **void** main(String[] args) {  //测试类创建笔记本对象  Computer computer = **new** Computer();  //开机方法  computer.begin();    //创建鼠标对象,调用使用USB设备方法  Mouse mouse = **new** Mouse();  computer.useUSB(mouse);    //创建键盘对象,调用使用USB设备方法  Keyboard keyboard = **new** Keyboard();  computer.useUSB(keyboard);    //关机方法  computer.over();  }  } |

# API练习

## 需求

键盘录入一个字符串，要求删除该字符串中的所有java字符串（最终的字符串中不能包含java），要求打印删除后的结果以及删除了几个java字符串

比如键盘录入："java woaijava,i like jajavava,i enjoy java"

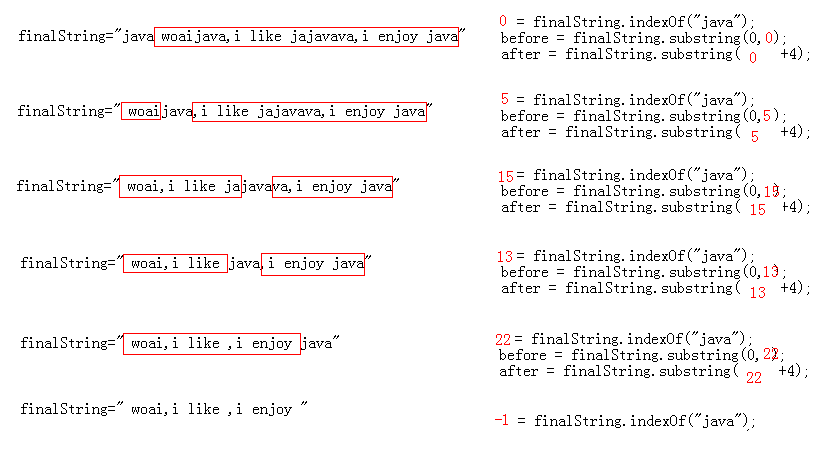
程序输出结果：

原字符串："java woaijava,i like jajavava,i enjoy java"中

总共包含：5个java删除java后的字符串为：" woai,i like ,i enjoy "

## 需求分析

截取过程：



## 需求实现

### 案例代码二:

|  |
| --- |
| **package** com.igeek\_01;  **import** java.util.Scanner;  /\*\*  \* **@ClassName**: StringTest  \* **@Description**: API练习题，字符串操作  \* **@date** 2018年2月2日 下午1:52:18  \* Company www.igeekhome.com  \*  \* 需求：  \* 键盘录入一个字符串，要求删除该字符串中的所有java字符串（最终的字符串中不能包含java），要求打印删除后的结果以及删除了几个java字符串  \* 比如键盘录入："java woaijava,i like jajavava,i enjoy java"  \* 程序输出结果：  \* 原字符串："java woaijava,i like jajavava,i enjoy java"中  \* 总共包含：5个java，删除java后的字符串为：" woai,i like ,i enjoy "  \*  \*====================================================================  \* 分析：  \* 1、使用键盘录入Scanner接收一个初识被判断的字符串，用变量initString接收  \* 2、定义变量，用于记录每次变化的那个字符串，即最终的字符串finalString  \* 3、定义变量，记录java的个数  \* 4、  \* 定义变量，记录java是否存在于字符串中，用索引表示(indexOf方法)  \* while循环删除：  \* 条件：  \* 为只要字符串中没有java字符串即不再循环  \* 如果索引不为-1，就说明要删除字符串中的java；  \* 如果索引为-1，就说明已经不再包含java了，循环终止；  \*  \* 循环删除过程：  \* 由于进入了循环，所以判断字符串中有一个java，个数计数器+1  \* 将java前的字符串截取记录(substring方法(0,索引i),从0到i)  \* 将java后的字符串截取记录(substring方法(索引i+4),所i+4到最后)  \* 将java前后的字符串再拼接成新的字符串  \* 再判断新字符串中是否存在java，使用indexOf方法重新获取索引  \*  \* 5、打印结果：  \* 打印原字符串  \* 打印java个数  \* 打印不包含java的新字符串  \*/  **public** **class** StringTest {  **public** **static** **void** main(String[] args) **throws** InterruptedException {  //1、使用键盘录入Scanner接收一个初识被判断的字符串，用变量initString接收  Scanner scanner = **new** Scanner(System.***in***);  //初始化数据  String initString = scanner.nextLine();    //2、定义变量，用于记录每次变化的那个字符串，即最终的字符串finalString  String finalString = initString;    //3、定义变量，记录java的个数  **int** javaNumber=0;    //4、循环过程  //定义变量，记录java是否存在于字符串中，用索引表示(indexOf方法)  **int** index = finalString.indexOf("java");    //条件：  //为只要字符串中没有java字符串即不再循环  //如果索引不为-1，就说明要删除字符串中的java；  //如果索引为-1，就说明已经不再包含java了，循环终止；  **while**(index!=-1){  System.***out***.println(index);    //加入语句，让循环每秒执行一次  Thread.*sleep*(1000);    //反复删除  //由于进入了循环，所以判断字符串中有一个java，个数计数器+1  javaNumber++;    //将java前的字符串截取记录(substring方法(0,索引i),从0到i)  String before = finalString.substring(0, index);    //将java后的字符串截取记录(substring方法(索引i+4),所i+4到最后)  String after = finalString.substring(index+4);    //将java前后的字符串再拼接成新的字符串  finalString = before+after;    //再判断新字符串中是否存在java，使用indexOf方法重新获取索引  index = finalString.indexOf("java");    System.***out***.println(finalString);  }    //5、打印结果  //格式  //原字符串："java woaijava,i like jajavava,i enjoy java"中  //总共包含：5个java，删除java后的字符串为：" woai,i like ,i enjoy "  System.***out***.println("============================================");    System.***out***.println("原字符串："+initString+"中");  System.***out***.println("总共包含："+javaNumber+"个java，删除java后的字符串为："+finalString);  }  } |

# 集合练习

## 需求:

分析以下需求，并用代码实现：

(1)定义一个学生类Student，属性：姓名(String name)、班级班号(String class\_number)、分数(double score)

(2)初始化数据将若干Student对象存入List集合

(3)以班级为单位,使用Map存储所有该班学生

(4)统计每个班级的总分和平均分

可选解法提示:

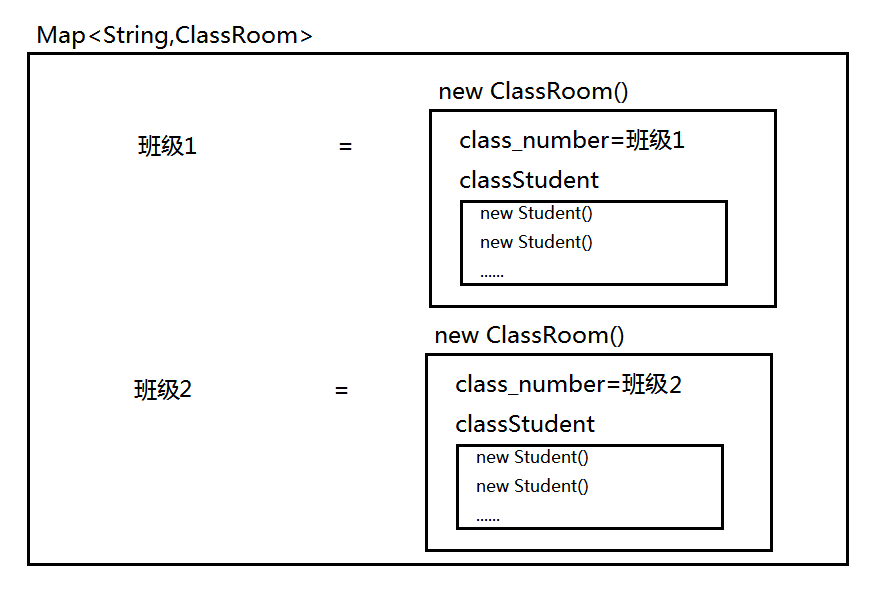
a.采用面向对象的思想(组合)

b.不推荐使用Map<String,List<Student>>

c.推荐使用Map<String，ClassRoom>

## 需求分析:

保存的结构图:



## 需求实现:

### 案例代码三:

|  |
| --- |
| **package** com.igeek\_01;  **import** java.util.ArrayList;  **import** java.util.Collections;  **import** java.util.HashMap;  **import** java.util.Set;  /\*\*  \* **@ClassName**: ListTest  \* **@Description**: 集合练习题  \* **@date** 2018年2月2日 下午2:53:15  \* Company www.igeekhome.com  \*  \* 分析以下需求，并用代码实现：  \* (1)定义一个学生类Student，属性：姓名(String name)、班级班号(String class\_number)、分数(double score)  \* (2)初始化数据将若干Student对象存入List集合  \* (3)以班级为单位,使用Map存储所有该班学生  \* map.put("班级1",班级1的学生);  \* map.put("班级2",班级2的学生);  \* ......  \* (4)统计每个班级的总分和平均分  \*  \* 可选解法提示:  \* a.采用面向对象的思想(组合)  \* b.不推荐使用Map<String,List<Student>>  \* c.推荐使用Map<String，ClassRoom>  \*  \*=========================================================  \* 分析：  \* 1、定义Student类  \* 2、创建多个Student对象，放到集合中，数据一共有3个班级，班级1、班级2、班级3，作为初始数据  \* 3、定义班级类：  \* 班号  \* 该班所有的学生  \*  \* 将该班级所有学生以Map<String,ClassRoom>的方式，存储到Map集合中  \* 定义变量，记录Map集合，用于存储所有班级(这里简化需求，只有三个班级，班级1、班级2、班级3)  \* 定义三个ArrayList，记录三个班级对象的所有学生信息  \* 迭代所有学生初始化数据集合，依次获取到每一个同学  \* 判断每个学生是班级1、班级2还是班级3，将该学生放到对应的ArrayList集合中  \* 创建三个ClassRoom对象  \* 再将三个班级加入到对应的Map集合中  \* 4、  \* 遍历Map集合，分别拿到每个班级的班级对象  \* 获取班级对象中所有学生的列表  \* 定义变量，记录总分  \* 定义变量，记录平均分  \* 循环遍历列表，累加总分  \* 计算平均分  \* 打印数据  \*/  **public** **class** ListTest {  **public** **static** **void** main(String[] args) {  //初始化数据  Student s = **new** Student("Jack", "班级1", 89);  Student s2 = **new** Student("Jack", "班级2", 29);  Student s3 = **new** Student("Jack", "班级1", 39);  Student s4 = **new** Student("Jack", "班级3", 49);  Student s5 = **new** Student("Jack", "班级2", 59);  Student s6 = **new** Student("Jack", "班级1", 50);  Student s7 = **new** Student("Jack", "班级1", 88);    //将若干Student对象存入List集合  ArrayList<Student> allStudents = **new** ArrayList<Student>();  Collections.*addAll*(allStudents, s,s2,s3,s4,s5,s6,s7);    //将该班级所有学生以Map<String,ClassRoom>的方式，存储到Map集合中  HashMap<String, ClassRoom> allClassMap = **new** HashMap<String, ClassRoom>();    //定义三个ArrayList，记录三个班级对象的所有学生信息  ArrayList<Student> class1Students = **new** ArrayList<Student>();  ArrayList<Student> class2Students = **new** ArrayList<Student>();  ArrayList<Student> class3Students = **new** ArrayList<Student>();    //迭代所有学生初始化数据集合，依次获取到每一个同学  **for** (Student thisStudent : allStudents) {  //判断每个学生是班级1、班级2还是班级3，将该学生放到对应的ArrayList集合中  **if**("班级1".equals(thisStudent.getClassNumber())){  class1Students.add(thisStudent);  }**else** **if**("班级2".equals(thisStudent.getClassNumber())){  class2Students.add(thisStudent);  }**else** **if**("班级3".equals(thisStudent.getClassNumber())){  class3Students.add(thisStudent);  }  }    //创建三个ClassRoom对象  ClassRoom class1 = **new** ClassRoom("班级1", class1Students);  ClassRoom class2 = **new** ClassRoom("班级2", class2Students);  ClassRoom class3 = **new** ClassRoom("班级3", class3Students);    //再将三个班级加入到对应的Map集合中  allClassMap.put("班级1", class1);  allClassMap.put("班级2", class2);  allClassMap.put("班级3", class3);      //遍历Map集合，分别拿到每个班级的班级对象  Set<String> classNumbers = allClassMap.keySet();    //迭代班级班号集合  **for** (String classNumber : classNumbers) {  //根据班级班号获取班级对象  ClassRoom classRoom = allClassMap.get(classNumber);  //调用方法，求总分与平均分  *getClassInfo*(classRoom);  }  }    /\*\*  \* **@Title**: getClassInfo  \* **@Description**: 接收一个班级，打印出该班级的总分与平均分  \* **@param** classRoom  \*/  **public** **static** **void** getClassInfo(ClassRoom classRoom){  //获取班级对象中所有学生的列表  ArrayList<Student> classStudent = classRoom.getStudentsList();  //定义变量，记录总分  **double** sum=0;  //定义变量，记录平均分  **double** avg=0;  //循环遍历列表，累加总分  **for** (Student student : classStudent) {  sum+=student.getScore();  }  //计算平均分  avg=sum/classStudent.size();  //打印数据  System.***out***.println(classRoom.getClassNumber()+"的总分为："+sum+",平均分为："+avg);  }  } |

|  |
| --- |
| **package** com.igeek\_01;  /\*\*  \* **@ClassName**: Student  \* **@Description**: 学生类  \* **@date** 2018年2月2日 下午3:33:23  \* Company www.igeekhome.com  \*  \* 定义一个学生类Student，属性：姓名(String name)、班级班号(String class\_number)、分数(double score)  \*/  **public** **class** Student {  /\*\*  \* **@Fields** name : 姓名  \*/  **private** String name;  /\*\*  \* **@Fields** classNumber : 班级班号  \*/  **private** String classNumber;  /\*\*  \* **@Fields** score : 分数  \*/  **private** **double** score;  /\*\*  \* **@Title**: Student  \*/  **public** Student() {  **super**();  }  /\*\*  \* **@Title**: Student  \* **@param** name  \* **@param** classNumber  \* **@param** score  \*/  **public** Student(String name, String classNumber, **double** score) {  **super**();  **this**.name = name;  **this**.classNumber = classNumber;  **this**.score = score;  }  /\*\*  \* **@return** the name  \*/  **public** String getName() {  **return** name;  }  /\*\*  \* **@param** name the name to set  \*/  **public** **void** setName(String name) {  **this**.name = name;  }  /\*\*  \* **@return** the classNumber  \*/  **public** String getClassNumber() {  **return** classNumber;  }  /\*\*  \* **@param** classNumber the classNumber to set  \*/  **public** **void** setClassNumber(String classNumber) {  **this**.classNumber = classNumber;  }  /\*\*  \* **@return** the score  \*/  **public** **double** getScore() {  **return** score;  }  /\*\*  \* **@param** score the score to set  \*/  **public** **void** setScore(**double** score) {  **this**.score = score;  }  } |

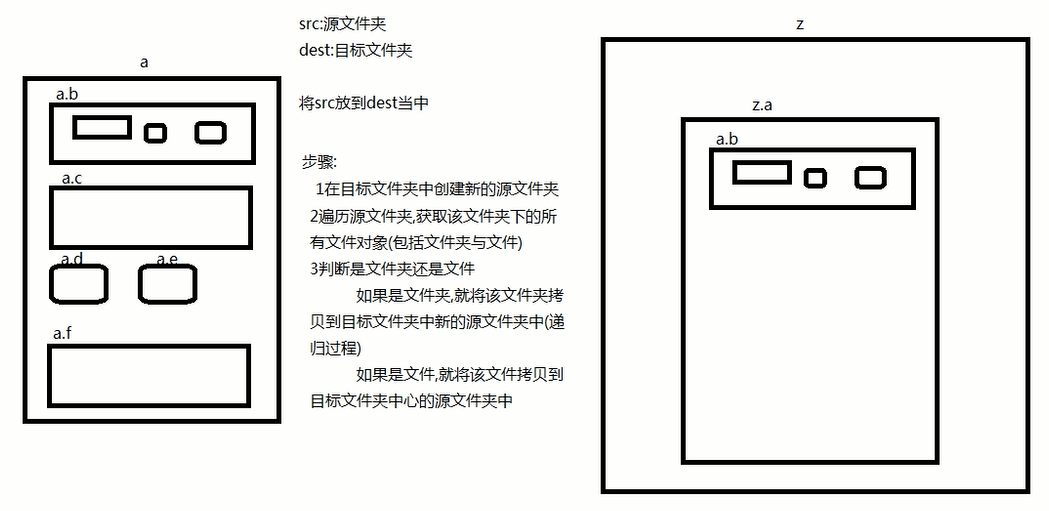
|  |
| --- |
| **package** com.igeek\_01;  **import** java.util.ArrayList;  /\*\*  \* **@ClassName**: ClassRoom  \* **@Description**: 班级类  \* **@date** 2018年2月2日 下午3:39:15  \* Company www.igeekhome.com  \*  \* 班级类：班级班号、学生列表  \*/  **public** **class** ClassRoom {  /\*\*  \* **@Fields** classNumber : 班级班号  \*/  **private** String classNumber;  /\*\*  \* **@Fields** students : 学生列表  \*/  **private** ArrayList<Student> studentsList;  /\*\*  \* **@Title**: ClassRoom  \*/  **public** ClassRoom() {  **super**();  }  /\*\*  \* **@Title**: ClassRoom  \* **@param** classNumber  \* **@param** studentsList  \*/  **public** ClassRoom(String classNumber, ArrayList<Student> studentsList) {  **super**();  **this**.classNumber = classNumber;  **this**.studentsList = studentsList;  }  /\*\*  \* **@return** the classNumber  \*/  **public** String getClassNumber() {  **return** classNumber;  }  /\*\*  \* **@param** classNumber the classNumber to set  \*/  **public** **void** setClassNumber(String classNumber) {  **this**.classNumber = classNumber;  }  /\*\*  \* **@return** the studentsList  \*/  **public** ArrayList<Student> getStudentsList() {  **return** studentsList;  }  /\*\*  \* **@param** studentsList the studentsList to set  \*/  **public** **void** setStudentsList(ArrayList<Student> studentsList) {  **this**.studentsList = studentsList;  }    } |

# IO递归练习

## 需求:

将一个文件夹中的内容(包含子文件夹中的所有内容)复制到指定目的地

## 需求分析:



## 需求实现:

### 案例代码四:

|  |
| --- |
| **package** com.igeek\_01;  **import** java.io.File;  **import** java.io.FileInputStream;  **import** java.io.FileOutputStream;  **import** java.io.IOException;  /\*\*  \* **@ClassName**: IOTest  \* **@Description**: IO练习题  \* **@date** 2018年2月2日 下午4:09:51  \* Company www.igeekhome.com  \*  \* 需求:  \* 完成多级文件夹复制功能(不允许使用CommonsIO)  \*  \* 分析见图:多级文件夹复制.png  \*/  **public** **class** IOTest {  **public** **static** **void** main(String[] args) {    //给出要复制的源文件夹与目标文件夹  File src = **new** File("a");  File dest = **new** File("z");    *copyDir2Dir*(src, dest);  }  /\*\*  \* **@Title**: copyDir2Dir  \* **@Description**: 将源文件夹src复制到目标文件夹dest  \* **@param** src  \* **@param** dest  \*/  **public** **static** **void** copyDir2Dir(File src,File dest) { |

重点和总结

1、面向对象的回顾

2、常用API的回顾

3、集合工具类的回顾

4、IO操作及递归的回顾