**Java课程上机报告**

## 上机题目：Java高级语言特征

张俊华 16030199025 (组长)

李金鑫 16030199026

李天浩 16030199027

**上机时间：2017/04/08 08:00-12:00**

**地点：EIII-204**

1. **小组名单**

|  |  |  |
| --- | --- | --- |
| 学号 | 姓名 | 工作 |
| 16030199025 | 张俊华 | 完成程序、练习例程、调试、小组讨论 |
| 16030199026 | 李金鑫 | 完成程序、练习例程、调试、小组讨论 |
| 16030199027 | 李天浩 | 完成程序、练习例程、调试、小组讨论 |

**二、题目**

**实验目标：**

熟悉Java的泛型；了解Java的泛型特点；初步掌握Java的泛型编程方法。

**实验要求：**

1.**练习PPT中7，21，31，53，64，70，74，79，81，83，94，99页的小练习，尝试对小练习中各部分进行修改，并观察修改后的执行效果。**

2.编写一个程序，程序提供记事本功能：

（1）构建记事本类，该类能存储不定数量的记录；能获得已经存储的记录数量；能追加记录；能展示已经存储的全部记录或其中任何一条记录；能删除已经存储的全部记录或其中任何一条记录。

（2）构建测试类，该类实现与用户的交互，向用户提示操作信息，并接收用户的操作请求。

程序应具有良好的人机交互性能，即：程序应向用户提示功能说明，并可根据用户的功能选择，执行对应的功能，并给出带详细描述信息的最终执行结果。

**三、题目分析**：

**本实验要求编写一个程序，实现以下功能：**

1. 根据用户输入储存记录

2. 能对记录进行追加、展示、删除操作

3. 良好的交互

1. 根据用户输入储存记录：

Java的ArrayList类采用可变大小的数组实现了List接口。除了实现List接口，该类还提供了访问数组大小的方法，ArrayList对象会随着元素的增加其容积自动扩大。这些特性适用于题目要求，故使用ArrayList储存记录。

但是，直接使用ArrayList并不是最佳选择，本题应该构建一个类对ArrayList进行封装，避免其他类对ArrayList直接调用。提高程序的安全性与拓展性，减少不同类之间的耦合。

1. 对记录的追加、展示、删除

构建NotesBox类，该类含有对ArrayList进行操作的方法，其他类通过对NotesBox方法的调用，实现对记录的操作

1. 实现良好人机交互的用户界面：

构建界面类，通过System.out.println向用户输出提示信息，并通过Scanner类接收用户输入。

**四、程序实现：**

张俊华：

**1．实验环境：**

IntelliJ IDEA 2017.1.2

Build #IU-171.4249.39, built on April 25, 2017

JRE: 1.8.0\_112-release-736-b16 amd64

JVM: OpenJDK 64-Bit Server VM by JetBrains s.r.o

Windows 10 10.0

**2．实现过程：**

在开始编写程序前先对程序的整体结构和各个类之间的关系进行设计。定义三个类：

NotesBox：对ArrayList进行操作，读写数据

Actions：程序功能和逻辑的实现

UserInterface：与用户交互，包括控制台的输入与输出

在NotesBox中编写了以下方法：

public static int getListSize() {  
 */\*\*  
 \*返回notes个数  
 \*****@param*** *\*****@return*** *size 个数  
 \*****@exception*** *\*****@author*** *张俊华 16030199025  
 \*/* return *notesList*.size();  
}  
  
public static int addNote(int index, String note) {  
 */\*\*  
 \*写入一个note  
 \*****@param*** *note 需要写入的字符串  
 \*****@return*** *index 写入位置  
 \*****@exception*** *\*****@author*** *张俊华 16030199025  
 \*/* if (index == -1)  
 *notesList*.add(note);  
 //序号为-1则直接末尾追加  
 else  
 *notesList*.add(index, note);  
 //否则插入到指定序号前  
 return *notesList*.indexOf(note);  
 //返回这个note在list中的位置  
  
  
}  
  
public static String getNote(int index) {  
 */\*\*  
 \*获取序号为index的note  
 \*****@param*** *index 序数  
 \*****@return*** *\*****@exception*** *\*****@author*** *张俊华 16030199025  
 \*/* return *notesList*.get(index);  
}  
  
public static void delNote(int index) {  
 */\*\*  
 \*删除指定序号的note  
 \*****@param*** *index 序数  
 \*****@return*** *\*****@exception*** *\*****@author*** *张俊华 16030199025  
 \*/  
 notesList*.remove(index);  
}  
  
public static void delAllNote() {  
 */\*\*  
 \*清空笔记本  
 \*****@param*** *\*****@return*** *\*****@exception*** *\*****@author*** *张俊华 16030199025  
 \*/  
 notesList*.clear();  
}

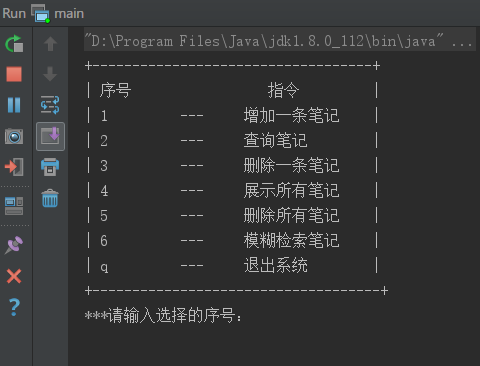
在Actions中实现了记事本的各种功能，比如addNote、delNote、showNote等，这些方法调用UserInterface中的方法和NotesBox中的方法对用户的输入进行处理和响应。

例如：删除一条笔记

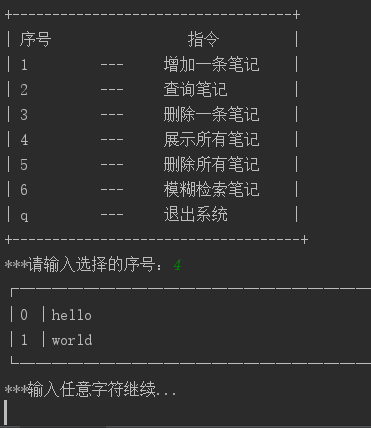
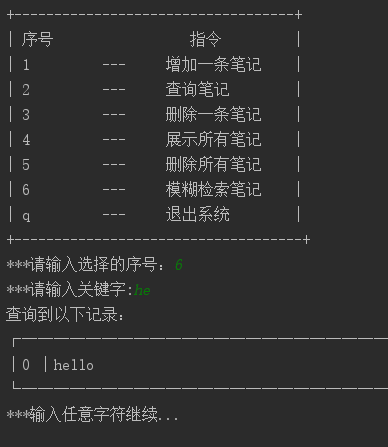
public static void delNote() {  
 */\*\*  
 \*删除一条note  
 \*****@param*** *\*****@return*** *\*****@exception*** *\*****@author*** *张俊华 16030199025  
 \*/* Prompt.*TipDelNote*(1);  
 int index = ScanTerminal.*ScanIndex*();  
 //获取删除序号  
 Prompt.*TipDelNote*(2);  
 PrintTerminal.*PrintNote*(NotesBox.*getNote*(index));  
 Prompt.*TipDelNote*(3);  
 //询问是否删除  
 if (ScanTerminal.*ScanChoise*().equals("y")) {  
  
 NotesBox.*delNote*(index);  
 Prompt.*TipDelNote*(4);  
 }  
 //执行删除操作  
 else {  
 Prompt.*TipDelNote*(5);  
 }  
 //取消操作  
}

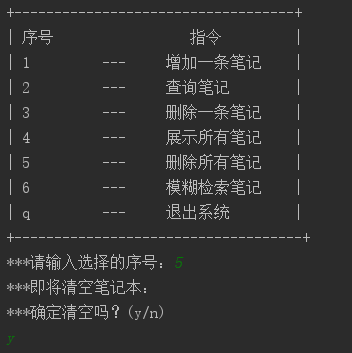
程序执行时发现，当用户请求的序号超出了ArrayList范围时，Java会抛出异常并终止运行，为了防止用户错误的输入导致程序崩溃，改进代码，加入异常处理：

public static boolean userChoise(){  
 */\*\*  
 \*获取用户想执行的功能  
 \*****@param*** *\*****@return*** *boolean 是否进行了选择  
 \*****@exception*** *\*****@author*** *张俊华 16030199025  
 \*/* Prompt.*TipShowMenu*();  
 //打印菜单  
 String choise = ScanTerminal.*ScanChoise*();  
 try {  
 if (choise.equals("1")) {  
 Actions.*addNote*();  
 } else if (choise.equals("2")) {  
 Actions.*showNote*();  
 } else if (choise.equals("3")) {  
 Actions.*delNote*();  
 } else if (choise.equals("4")) {  
 Actions.*showAllNote*();  
 } else if (choise.equals("5")) {  
 Actions.*delAllNote*();  
 } else if (choise.equals("6")) {  
 Actions.*searchNote*();  
 } else if (choise.equals("q")) {  
 return false;  
 }  
 } catch (Exception ex) {  
 Prompt.*TipWrong*(1);  
 }  
 return true;  
}

运行结果：  






# 

# 李天浩:

如题目分析

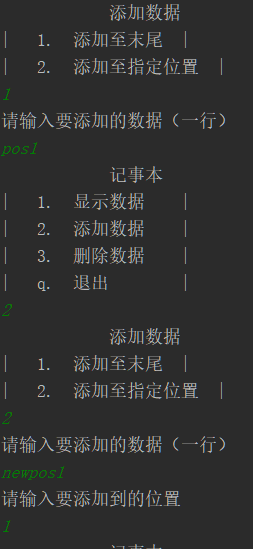
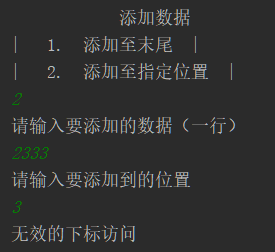
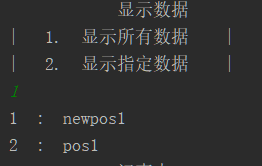
获得记录数量就简单调用下

向指定位置插入和删除指定位置我首先对用户输入的位置进行了判断，看其是不是一个有效的位置，这样能够避免越界等异常发生

遍历使用了迭代器操作

构建用户界面这部分还是不熟练，使用OOP的写法来写交互感觉比较困难。

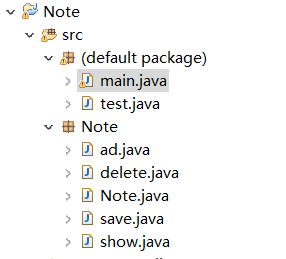
# 运行结果

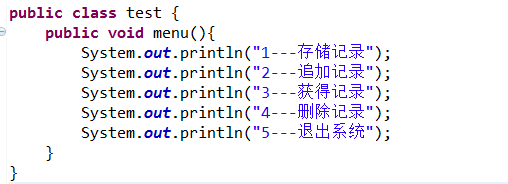
 

**李金鑫:**

**实现过程：**

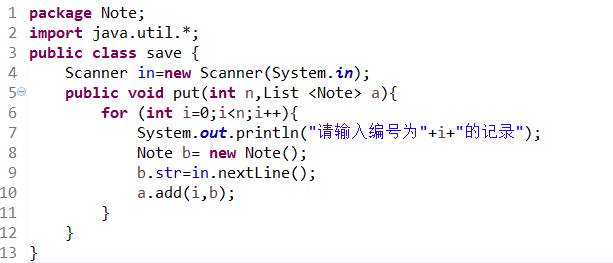
程序的框架如下，其中关于记事本的操作类全部放在Note包内。



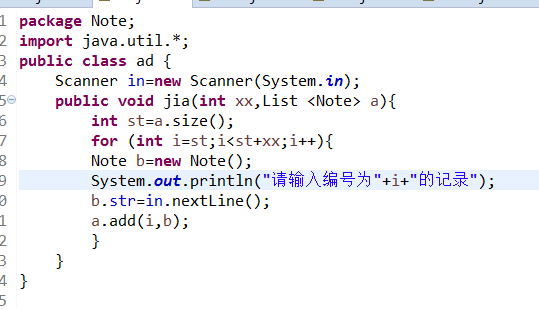
主界面如下，每次实现一个功能都把参数传到与之相关的类中。



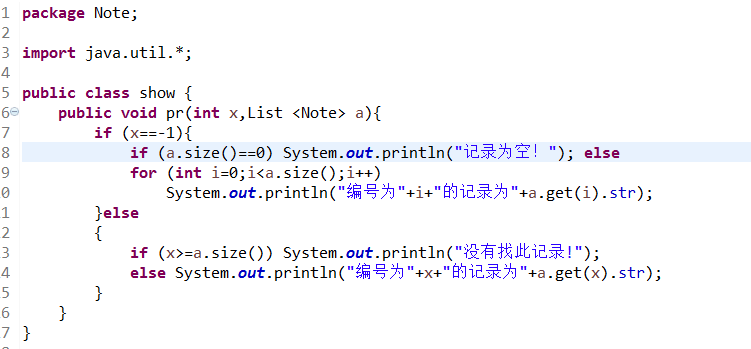
保存记录用了.add方法，并且可以选择保存的数量。加入了标记，如果原记录未空，则不可以再重新保存，应该追加保存或删除原有的记录重新存储。



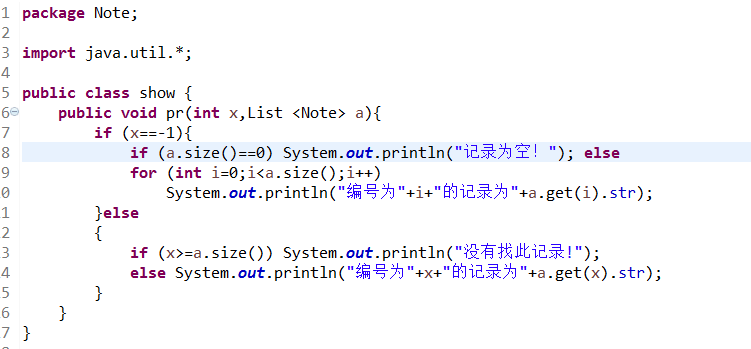
追加记录也是用.add方法，并且从list.size()处存储。



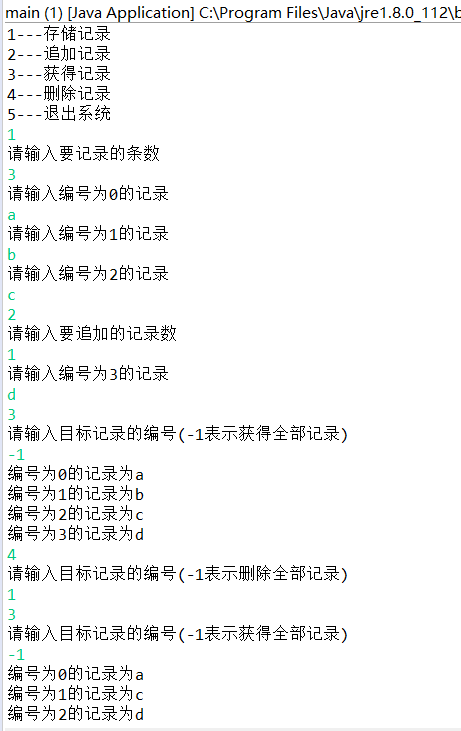
展示记录加入了判断标记，分别是list为空和要查的记录不存在时的警告。



删除操作如果全部删除，调用.clear()方法，单个删除用了.remove方法。



**运行结果：**



二．小组讨论内容：

**张俊华**：

首先观察组内同学李金鑫的程序，其优点在于优化了交互逻辑，在题目要求的基础上，增加了批量添加笔记的功能。

不足之处在于：程序中类的划分较为混乱，将Add、Del等动作分别包装成类，而这些应该属于Note类的方法，不应该单独成类。观察Note类，里面只声明了一个字符串，这个类没有存在的必要。

再观察其main类中的实现，发现ArrayList定义在main方法内部，且对外未提供任何对链表元素修改的接口，main方法直接与控制台交互，可拓展性差。对各种Add、Del类实例化后调用，基本是类似于c语言函数，没有体现出面向对象的设计思路。

再观察组内同学李天浩的程序，其定义了Container、Main、UI三个类，其中Container实现了数据的增、删、改、查的操作，Main实现了与控制台的交互，UI类包含了程序固有的交互提示。各种类的职责和关系较为明确。ArrayList被private权限保护，避免了数据的直接访问，Container预留了各种接口，基本能满足程序拓展的需要。整个程序代码精简，功能完备，无明显错误。

程序可以考虑完善与用户的交互，获得更好体验，其次，补充一些必要的注释会增强代码的可读性。

**李金鑫：**

张俊华：完整的用java泛型完成了题目要求，并且有很多注释，创新的写了模糊搜索功能，唯一不足的是程序显得不够简洁，有的功能的实现可以写的更简单易读。

李天浩：结构很简洁，完整的用java泛型完成了题目要求，设定了越界警告。不过注释缺乏，整个程序没有突出体现面向对象的思想。

**李天浩：**

审计了小组内另外两名同学的代码，查看了他们程序的运行情况

## 李金鑫

他的程序中，加入了导入功能，在进入时就可以大量输入数据而不用让用户反复调用添加操作，这一点是很好的。

美中不足的是在他的代码里大量使用了类，但是我个人认为对一个具体的东西进行的系列操作都应该是其类里的方法。我认为类应该是一个具体的东西，而不是一些操作。

## 张俊华

这位同学的代码里也用了很多类，但这些类分工明确，并没有冗余感。

UI设计很美观，代码里注释也很多，足见其用心。

使用try 来处理异常的方法也很通用。

有很多值得学习和借鉴的地方。

**六、个人总结：**

**张俊华：**

这次编写代码前，我先对程序的结构和类的功能进行了充分的考虑，事实证明，良好的程序构思使得编写程序的过程更加顺利，避免了许多无用代码，编写过程中也不用重新调整思路。

这次的不足之处在于：程序内大量使用了static关键字，过多的静态类降低了程序的扩展性。

**李金鑫：**

通过本次上机更深刻的理解了Arraylist的用法和面向对象的思想。熟悉了Java泛型，初步掌握了泛型编程方法。通过小组讨论认识到了程序的不足，关于Note的操作应该放在一个类中，而不是开很多类每个类只写一个方法，以后的编程中要注意这一点。

**李天浩**

通过这次上机，我熟悉了Java的泛型，了解了Java的泛型特点，初步掌握了Java的泛型编程方法。

由于对C++的 Standard Template Library 比较熟练，而这两门语言的容器又很相近，类比一下掌握起来还是比较轻松的。但是JAVA似乎没有运算符（操作符）重载有关的写法，我觉得如果有 random access iterator 或者随机访问时间复杂度比较小的容器如果有operator[]的话应该会很方便吧。对一些不支持随机访问的容器用get方法的话，可能会造成时间的巨大浪费。

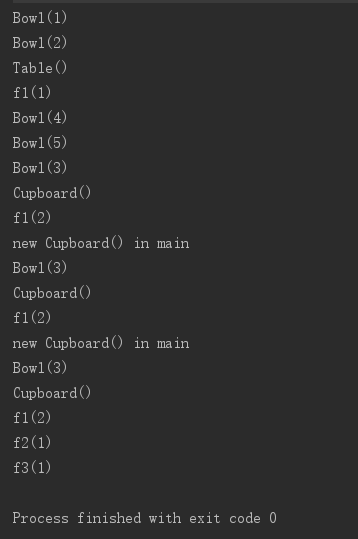
程序设计方面，对给出的具体任务能较好的用OOP的思想解决，但是对于和用户的交互一类还是很难有好的OOP思路，这一点还需要提高

**七、ppt小练习**

**练习1：**

* **静态变量的创建与实例对象无关**
* **只在系统加载其所在类时分配空间并初始化，且在创建该类的实例对象时不再分配空间**
  + **什么时候加载其所在类？**
    - **运行到不得不加载该类的时候**
  + **什么是“不得不加载该类的时候”？**
    - **即将创建该类的第一个对象时**
    - **首次使用该类的静态方法或静态变量时**
    - **加载一个类的子类之前要先加载其父类**
* */\*\*  
   \* Created by 张俊华 on 2017/5/15.  
   \*  
   \** ***@author*** *张俊华.  
   \** ***@Time*** *2017/5/15 18:07.  
   \*/*import static java.lang.System.*out*;  
    
   class Bowl {  
   Bowl(int i) {  
   *out*.println("Bowl(" + i + ")");  
   }  
    
   void f1(int i) {  
   *out*.println("f1(" + i + ")");  
   }  
   }  
    
   class Table {  
   static Bowl *bowl1* = new Bowl(1);  
    
   Table() {  
   *out*.println("Table()");  
   *bowl2*.f1(1);  
   }  
    
   void f2(int i) {  
   *out*.println("f2(" + i + ")");  
   }  
    
   static Bowl *bowl2* = new Bowl(2);  
   }  
    
   class Cupboard {  
   Bowl bowl3 = new Bowl(3);  
   static Bowl *bowl4* = new Bowl(4);  
    
   Cupboard() {  
   *out*.println("Cupboard()");  
   *bowl4*.f1(2);  
   }  
    
   void f3(int i) {  
   *out*.println("f3(" + i + ")");  
   }  
    
   static Bowl *bowl5* = new Bowl(5);  
    
   }  
    
   public class StaticInitialization {  
   public static void main(String[] args) {  
   *out*.println("new Cupboard() in main");  
   new Cupboard();  
   *out*.println("new Cupboard() in main");  
   new Cupboard();  
   *table*.f2(1);  
   *cupboard*.f3(1);  
   }  
    
   static Table *table* = new Table();  
   static Cupboard *cupboard* = new Cupboard();  
   }

**输出结果：**

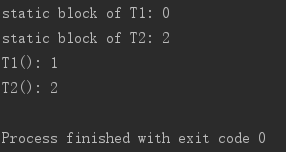


**练习2：**

**写下以下程序的输出：**

class T1 {  
 static int *s1* = 1;  
 static { System.*out*.println("static block of T1: " + T2.*s2*); }  
 T1() { System.*out*.println("T1(): " + *s1*); }  
}  
class T2 extends T1 {  
 static int *s2* = 2;  
 static { System.*out*.println("static block of T2: " + T2.*s2*); }  
 T2() { System.*out*.println("T2(): " + *s2*); }  
}  
public class InheritStaticInit {  
 public static void main(String[] args) {  
 new T2();  
 }  
}

**输出结果：**



**练习3：在成员变量中使用Final关键字**

* **空白final：若final成员变量声明时未赋初值，则在所属类的每个构造方法中都必须对该变量赋值**

class Poppet {  
 private int i;  
 Poppet(int ii) { i = ii; }  
}  
public class BlankFinal {  
 private final int i = 0; // 被初始化的final  
 private final int j; // 空白final  
 private final Poppet p; // 空白final引用  
  
 public BlankFinal() {  
 j = 1; // 初始化空白final  
 p = new Poppet(1); // 初始化空白final引用  
 }  
  
 public BlankFinal(int x) {  
 j = x; // 初始化空白final  
 p = new Poppet(x); // 初始化空白final引用  
 }  
  
 public static void main(String[] args) {  
 new BlankFinal();  
  
 new BlankFinal(47);  
 }  
}

**练习4：**

* **下列接口的定义中，哪些是正确的？**

**(1) interface Printable{**

**void print() {};**

**}**

**(2) abstract interface Printable{**

**void print();**

**}**

**(3) abstract interface Printable extends Interface1, Interface2{**

**void print() {};**

**}**

**(4) interface Printable{**

**void print();**

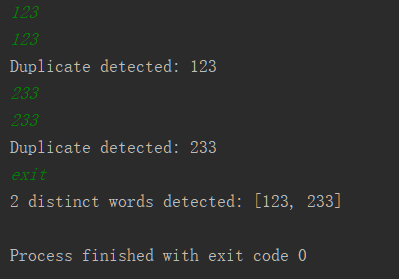
**}**

**解答：**

**接口（4）是正确的**

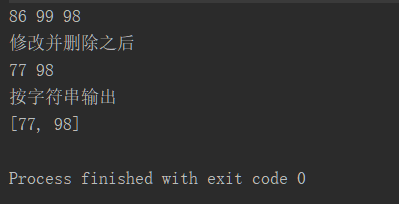
64

public class FindDups {  
 public static void main(String args[]) {  
 Scanner in = new Scanner(System.*in*);  
 Set<String> s = new HashSet<String>();  
 //其中重复的字符串将不能加入，并被打印输出  
 String op;  
 while (true) {  
 op=in.next();  
 if(op.equals("exit")){  
 break;  
 }  
 if (!s.add(op)){  
 System.*out*.println("Duplicate detected: " + op);  
 }  
 }  
 // 输出集合s的元素个数以及集合中的所有元素：  
 System.*out*.println(s.size() + " distinct words detected: " + s);  
 }  
  
}

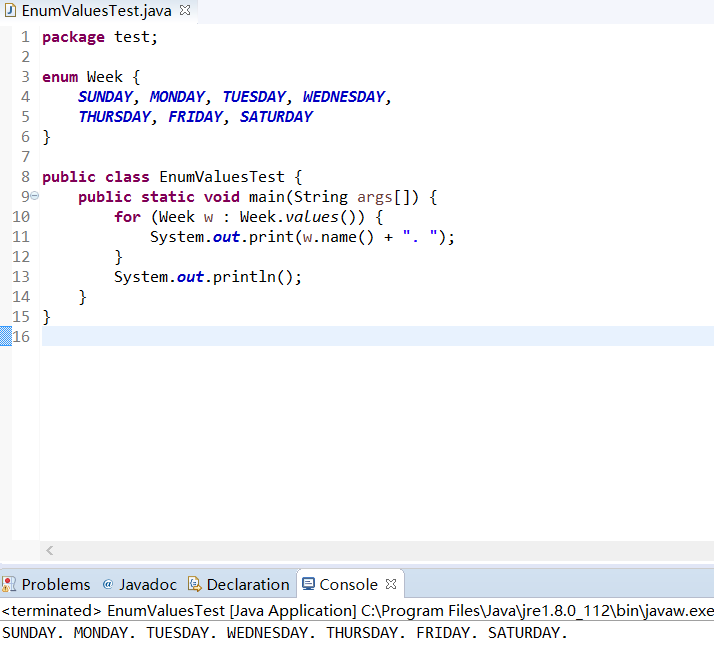
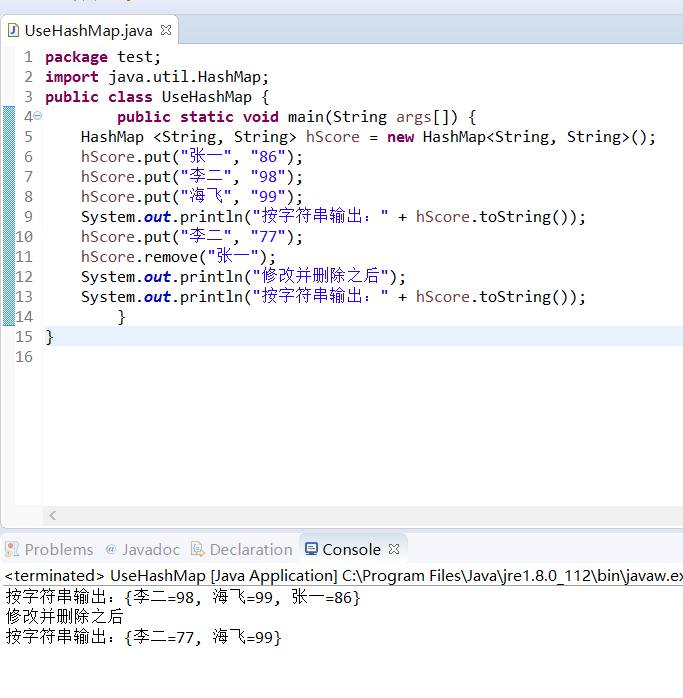


70

public class UseArrayList {  
 public static void main(String[] args) {  
 List<String> scores = new ArrayList<String>();  
 scores.add("86"); // 添加元素  
 scores.add("98"); // 添加元素  
 scores.add(1, "99"); // 插入元素  
 for (int i = 0; i < scores.size(); i++) {  
 System.*out*.print(scores.get(i) + " ");// 输出结果  
 }  
 scores.set(1, "77"); // 修改第二个元素  
 scores.remove(0); // 删除第一个元素  
 System.*out*.println("\n修改并删除之后");  
 for (int i = 0; i < scores.size(); i++) {  
 System.*out*.print(scores.get(i) + " ");  
 }  
 System.*out*.println(" \n按字符串输出\n" + scores.toString());  
 }  
}

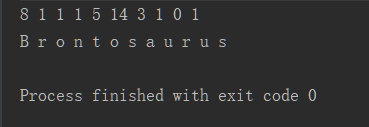


**PPT程序测试：**

****

74

import java.util.\*;  
public class QueueDemo {  
 public static void printQ(Queue queue) {  
 while (queue.peek() != null)  
 System.*out*.print(queue.remove() + " ");  
 System.*out*.println();  
 }  
  
 public static void main(String[] args) {  
 Queue<Integer> queue = new LinkedList<Integer>();  
 Random rand = new Random(47);  
 for (int i = 0; i < 10; i++)  
 queue.offer(rand.nextInt(i + 10));  
 *printQ*(queue);  
  
 Queue<Character> qc = new LinkedList<Character>();  
 for (char c : "Brontosaurus".toCharArray())  
 qc.offer(c);  
 *printQ*(qc);  
 }  
}



79

import java.util.\*;  
  
public class Freq {  
 public static void main(String args[]) {  
 String[] words = { "if", "it", "is", "to", "be", "it",  
 "is", "up", "to", "me", "to", "delegate" };  
 Integer freq;  
 Map<String, Integer> m = new TreeMap<String, Integer>();  
  
 for (String a : words) {//以(单词,词频)为键值对, 构造频率表  
 freq = m.get(a); // 获取指定单词的词频。  
 if (freq == null) { // 词频递增  
 freq = new Integer(1);  
 } else {  
 freq = new Integer(freq + 1);// .intValue()  
 }  
 m.put(a, freq); // 在Map中更改词频  
 }  
 System.*out*.println(m.size() + " distinct words detected:");  
 System.*out*.println(m);  
 }  
}