**《面向对象程序设计(Java)》**

**实验报告**

姓 名 石家晖

学 号 2019081238

学 院 软件工程学院

专 业 软件工程

年 级 2019级

指导教师 文立玉

**成都信息工程大学软件工程学院**

**2020年9月**

# 实验6：

1. **实验目的**

1.了解Set接口、List接口、Map接口及其主要实现类

2. 掌握ArrayList、 HashMap的使用

3. 掌握泛型类、泛型方法的创建和使用

4. 理解列表与映射、枚举和迭代 等集合类，能够创建并使用集合对象

1. **实验要求**

1. 实验在2学时内完成。

2. 1人1组独立完成。

1. **实验设备与环境**

JDK13, NEOVIM，Linux

1. **设计思路与具体实现**

1.

(1)设计思路

1) 创建一个只能容纳String对象名为names的ArrayList集合；

2）按顺序往集合中添加5个字符串对象：“张三”、“李四”、“王五”、“马六”、“赵七”；

3）对集合进行遍历，分别打印集合中的每个元素的位置与内容；

4）首先打印集合的大小，然后删除集合中的第3个元素，并显示删除元素的内容，然后再打印目前集合中第3个元素的内容，并再次打印集合的大小。

(2)实验步骤

①创建容纳String对象名为names的ArrayList集合

②使用.add方法添加元素及内容

③使用.size()方法控制遍历输出内容和位置

④使用.remove方法删除元素

⑤遍历删除元素后的集合

(3)关键代码

ArrayList<String> names = new ArrayList<String>();

names.add("张三");

names.add("李四");

names.add("王五");

names.add("马六");

names.add("赵七");

for (int i = 0;i < names.size();i++) {

System.out.println("位置:"+(i+1)+"内容:"+names.get(i));

}

System.out.println("当前集合大小:"+names.size());

System.out.println("删除元素的内容:"+names.remove(2));

System.out.println("当前第三个位置元素的内容:"+names.get(2));

System.out.println("当前集合大小:"+names.size());

System.out.println("遍历删除元素后的集合:");

for (int i = 0;i < names.size();i++) {

System.out.println("位置:"+(i+1)+"内容:"+names.get(i));

}

}

(4)程序测试过程

①编译运行程序

②输入测试用例

③检验结果是否与预期相符

(5)程序运行结果(运行效果)

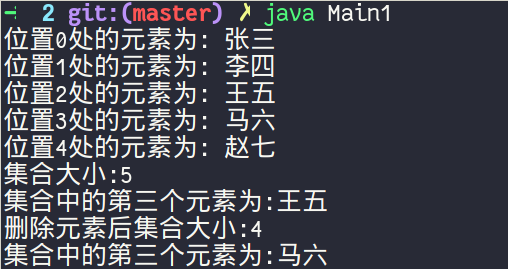


图6-1

2.

(1)设计思路

使用集合框架中的HashMap的键值对数据结构对面，每一个字母出现的位置进行储存，其中键为字母，值为ArrayList储存出现的位置。

(2)实验步骤

①新建主类CharIndex 用于实现题目操作

②定义类方法AddEntry用于将某字母出现的位置加入到对应的ArrayList中

③定义类构造方法负责将传入的字符串拆分以及运行②

④定义主方法用于接收命令行参数以及构造对象和操作对象，最后打印结果

(3)关键代码

import java.util.\*;  
public class Main2{  
 Map map = new HashMap();  
 public er(String s) {  
 for(int i = 0;i<s.length();++i)  
 add(s.charAt(i),i);  
 }  
 public void add(char c,int a) {  
 Character key = new Character(c); //获取key  
 List L = (List)map.get(key); //通过key找到value  
 //如果List为空,将key和l的映射放到Map中  
 if(L == null) {  
 L=new ArrayList();  
 map.put(key, L);  
 }  
 L.add(new Integer(a)); //把字符位置添加到List中  
 }  
 public static void main(String[] args) {  
 StringBuffer b = new StringBuffer();  
 for(int i = 0;i<args.length;++i) {  
 b.append(args[i]); //将字符添加到StringBuffer中  
 }  
 er er = new er(b.toString());  
 System.out.println(er.map); //输出结果  
 }  
}

(4)程序测试过程

①编译执行程序代码

②检查程序运行结果是否与预期符合

(5)实验结果(实验效果)

2020-11-22-234737_788x58_scrot

图6-2

1. **实验总结**

本次实验我通过使用List接口里的ArrayList集合，并通过add、get、size、remove等方法实现对集合的内容的添加、查找、遍历、删除等操作。此外，我了解了Java集合框架的基本概念和组成部分，深入的了解了Set接口、List接口、Map接口及其主要实现类，掌握了ArrayList、 HashMap的使用掌握泛型类、泛型方法的创建和使用。我还意识到了迭代器的使用的重要性，最重要的是，我能够根据实际问题的不同选用不同数据结构的集合实现类，对我来说收获颇多。

1. **附录（程序代码）**

import java.util.\*;

public class Main1{

public static void main (String[] args) {

//1) 创建一个只能容纳String对象名为names的ArrayList集合

ArrayList<String> names = new ArrayList<String>();

//2）按顺序往集合中添加字符串对象

names.add ("张三");

names.add ("李四");

names.add ("王五");

names.add ("马六");

names.add ("赵七");

//3）对集合进行遍历，分别打印集合中的每个元素的位置与内容

for (String i : names) //for-each语句遍历集合元素

System.out.println ("位置" + names.indexOf (i) + "处的元素为: " + i);

//4）首先打印集合的大小，然后删除集合中的第3个元素，并显示删除元素的内容

// 然后再打印目前集合中第3个元素的内容，并再次打印集合的大小。

System.out.println ("集合大小:" + names.size() ); //集合大小

System.out.println ("集合中的第三个元素为:" + names.get (2) ); //集合中的第三个元素

names.remove (names.get (3 - 1) ); //删除集合中的第三个元素

System.out.println ("删除元素后集合大小:" + names.size() ); //删除元素后集合大小

System.out.println ("集合中的第三个元素为:" + names.get (3 - 1) ); //删除元素后,集合中的第三个元素

}

}

2.

import java.util.\*;  
public class Main2{  
 Map map = new HashMap();  
 public er(String s) {  
 for(int i = 0;i<s.length();++i)  
 add(s.charAt(i),i);  
 }  
 public void add(char c,int a) {  
 Character key = new Character(c); //获取key  
 List L = (List)map.get(key); //通过key找到value  
 //如果List为空,将key和l的映射放到Map中  
 if(L == null) {  
 L=new ArrayList();  
 map.put(key, L);  
 }  
 L.add(new Integer(a)); //把字符位置添加到List中  
 }  
 public static void main(String[] args) {  
 StringBuffer b = new StringBuffer();  
 for(int i = 0;i<args.length;++i) {  
 b.append(args[i]); //将字符添加到StringBuffer中  
 }  
 er er = new er(b.toString());  
 System.out.println(er.map); //输出结果  
 }  
}