**每日作业卷答案**

**就业班JavaSE第2天**

传智播客.黑马程序员

基础题

### 练习一：集合框架

1. 请简述集合框架。

集合按照其存储结构可以分为两大类，分别是单列集合java.util.Collection和双列集合java.util.Map。

Collection：单列集合类的根接口，用于存储一系列符合某种规则的元素，它有两个重要的子接口，分别是java.util.List和java.util.Set。其中，List的特点是元素有序、元素可重复。Set的特点是元素无序，而且不可重复。List接口的主要实现类有java.util.ArrayList和java.util.LinkedList，Set接口的主要实现类有java.util.HashSet和java.util.TreeSet。

### 练习二：Collection集合统计元素出现次数

1. 给定以下代码，请定义方法listTest()统计集合中指定元素出现的次数，如"a": 2,"b": 2,"c" :1, "xxx":0。

|  |
| --- |
| 分析:   1. 定义一个listTest方法🡪需要参数(list,String s) 2. 定义一个变量🡪记录次数 3. 遍历集合,判断遍历出来的元素和我们指定的字符串是不是相等(内容相等) 4. 如果相等,变量++ |

Collection<String> list = new ArrayList<>();

list.add("a");

list.add("a");

list.add("b");

list.add("b");

list.add("c");

System.out.println("a:"+listTest(list, "a"));

System.out.println("b:"+listTest(list, "b"));

System.out.println("c:"+listTest(list, "c"));

System.out.println("xxx:"+listTest(list, "xxx"));

**public class** CollectionTest01{  
 **public static void** main(String[] args) {  
 Collection<String> list = **new** ArrayList<>();  
 list.add(**"a"**);  
 list.add(**"a"**);  
 list.add(**"b"**);  
 list.add(**"b"**);  
 list.add(**"c"**);  
 System.**out**.println(**"a:"**+listTest(list, **"a"**));  
 System.**out**.println(**"b:"**+listTest(list, **"b"**));  
 System.**out**.println(**"c:"**+listTest(list, **"c"**));  
 System.**out**.println(**"xxx:"**+listTest(list, **"xxx"**));  
 }  
  
 //定义方法统计集合中指定元素出现的次数  
 **public static int** listTest(Collection<String> list,String s){  
 //定义计数器，初始化为0  
 **int** count = 0;  
 //增强for遍历集合  
 **for** (String string : list) {  
 //判断传入方法的字符与遍历集合的是否一致  
 **if** (s.equals(string)) {  
 //如果一致，加1  
 count++;  
 }  
 }  
 **return** count;  
 }  
}

### 练习三：Collection集合数组转集合

1. 定义一个方法，要求此方法把int数组转成存有相同元素的集合(集合里面的元素是Integer)，并返回。()

|  |
| --- |
| 大概:  1.定义一个数组,存一些数  2.定义一个集合  3.一边遍历数组.一边调用add方法往集合中存 |

**public class** CollectionTest02 {  
 **public static void** main(String[] args) {  
 //定义int数组  
 **int**[] arr = {1,2,3,4,5};  
 ArrayList<Integer> list = listTest(arr);  
 System.**out**.println(list);  
 }  
  
 **public static** ArrayList<Integer> listTest(**int**[] arr) {  
 //定义集合  
 ArrayList<Integer> list = **new** ArrayList<Integer>();  
 //遍历数组，把元素依次添加到集合当中  
 **for** (**int** a : arr) {  
 list.add(a);  
 }  
 **return** list;  
 }  
}

### 练习四：Collection集合集合转数组

1. 定义一个集合，并把集合(集合里面的元素是Integer)转成存有相同元素的数组，并将结果输出在控制台。（可以使用Object[]数组类型接收转换的数组）

**public class** CollectionTest03 {  
 **public static void** main(String[] args) {  
 //定义集合,添加数据  
 ArrayList<Integer> list = **new** ArrayList<Integer>();  
 list.add(100);  
 list.add(200);  
 list.add(300);  
 //Object[] toArray()转换成一个Object数组  
 Object[] obj = list.toArray();  
 // 遍历数组  
 **for** (**int** i = 0; i < obj.**length**; i++) {  
 System.**out**.println(obj[i]);  
 }  
 }  
}

### 练习五：Collection集合contains()方法使用

1. 定义一个方法listTest(ArrayList<String> al, String s),要求使用contains()方法判断al集合里面是否包含s。

|  |
| --- |
| Al调用contains方法(s) |

**public class** CollectionTest04 {  
 **public static void** main(String[] args) {  
 //定义集合，添加数据  
 ArrayList<String> list = **new** ArrayList<String>();  
 list.add(**"itcast"**);  
 list.add(**"itheima"**);  
 list.add(**"java"**);  
 System.**out**.println(listTest(list,**"java"**));  
 }  
  
 **public static boolean** listTest(ArrayList<String> al, String s) {  
 //判断s是否在集合中存在,存在返回true，不存在返回false  
 **if** (al.contains(s)) {  
 **return true**;  
 }  
 **return false**;  
 }  
}

### 练习六：Collection集合isEmpty()方法的使用

1. 定义一个方法listTest(ArrayList<String> al), 要求使用isEmpty()判断al里面是否有元素。

**public class** CollectionTest05 {  
 **public static void** main(String[] args) {  
 //定义集合，添加数据  
 ArrayList<String> list = **new** ArrayList<String>();  
 list.add(**"1"**);  
 System.**out**.println(listTest(list));  
 }  
  
 **public static boolean** listTest(ArrayList<String> al) {  
 //判断al集合是否为空,为空返回true，不为空返回false  
 **if**(al.isEmpty()){  
 **return true**;  
 }  
 **return false**;  
 }  
}

### 练习七：简述迭代器的实现原理

1. 请简述迭代器的实现原理

当遍历集合时，首先通过调用集合的iterator()方法获得迭代器对象，然后使用hashNext()方法判断集合中是否存在下一个元素，如果存在，则调用next()方法将元素取出，否则说明已到达了集合末尾，停止遍历元素。

Iterator迭代器对象在遍历集合时，内部采用指针的方式来跟踪集合中的元素，在调用Iterator的next()方法之前，迭代器的索引位于第一个元素之前，不指向任何元素，当第一次调用迭代器的next方法后，迭代器的索引会向后移动一位，指向第一个元素并将该元素返回，当再次调用next方法时，迭代器的索引会指向第二个元素并将该元素返回，依此类推，直到hasNext方法返回false，表示到达了集合的末尾，终止对元素的遍历。

### 练习八：Collection集合返回首次出现索引

1. 定义一个方法listTest(ArrayList<Integer> al, Integer s)，要求返回s在al里面第一次出现的索引，如果s没出现过返回-1。

|  |
| --- |
| 步骤:   1. 定义方法listTest(ArrayList<Integer> al, Integer s)🡪需要返回值 2. 遍历集合,拿传递过来的s去和遍历出来的元素做比较,如果相等,马上将索引返回 |

**public class** CollectionTest06 {  
 **public static void** main(String[] args) {  
 //定义集合，添加数据  
 ArrayList<Integer> list = **new** ArrayList<Integer>();  
 list.add(1);  
 list.add(2);  
 list.add(3);  
 list.add(4);  
 list.add(5);  
 System.**out**.println(listTest(list, 5));  
 }  
  
 **public static int** listTest(ArrayList<Integer> al, Integer s) {  
 //遍历集合，获取元素，判断元素是否与s相等，相等返回索引  
 **for** (**int** i = 0; i < al.size(); i++) {  
 **if** (al.get(i).equals(s)) {  
 **return** i;  
 }  
 }  
 **return** -1;  
 }  
}

## 扩展题

### 练习九：Collection集合练习

1. (复杂，并不难)定义一个学生类Student，包含三个属性姓名、年龄、性别，创建三个学生对象存入ArrayList集合中。

A：遍历集合遍历输出。

B：求出年龄最大的学生，然后将该对象的姓名变为：小猪佩奇。

|  |
| --- |
| 分析:   1. 定义学生类 2. 创建三个学生对象,创建一个集合,将对象存到集合中🡪add(Student) 3. 遍历🡪将对象遍历出来,并调用getxxx() 4. 定义一个变量初始化为0 5. 遍历集合,判断我们这个变量是不是小于我们取出来的元素的年龄 6. 如果小,证明我们取出来的年龄大,我们就把大的给这个变量 7. 如果拿出最大的,直接用这个最大年龄的对象去调用setxxx(“小猪佩奇”) |

**public class** CollectionTest07 {  
 **public static void** main(String[] args) {  
 //定义集合，向集合中添加student对象  
 ArrayList<Student> list = **new** ArrayList<Student>();  
 list.add(**new** Student(**"张三"**, 23, **"男"**));  
 list.add(**new** Student(**"王五"**, 28, **"男"**));  
 list.add(**new** Student(**"李四"**, 25, **"男"**));  
 print(list);  
 System.**out**.println(**"--------------"**);  
 change(list);  
 System.**out**.println(**"--------------"**);  
 System.**out**.println(list);  
 }  
 //  
 **public static void** change(ArrayList<Student> list) {  
 //定义变量存放年龄  
 **int** a = 0;  
 //定义变量存放最大年龄的索引值  
 **int** index = 0;  
 //遍历集合获取年龄值，与a相比较  
 **for** (**int** i = 0; i < list.size(); i++) {  
 **if** (list.get(i).getAge() > a) {  
 //如果年龄大于a，记录次数  
 index = i;  
 //并把年龄的最大值赋予a  
 a = list.get(i).getAge();  
 }  
 }  
 System.**out**.println(**"年龄最大的学生是"** + list.get(index).getName());  
 //将年龄最大的学生姓名变为：小猪佩奇  
 list.get(index).setName(**"小猪佩奇"**);  
 }  
 //定义方法，遍历集合输出  
 **public static void** print(ArrayList<Student> list) {  
 **for** (Student student : list) {  
 System.**out**.println(student);  
 }  
 }  
}

### 练习十：Collection集合练习

1. 产生10个1-100的随机数，并放到一个数组中，把数组中大于等于10的数字放到一个list集合中，并打印到控制台。

|  |
| --- |
| 分析:   1. 定义一个数组 2. 创建Random对象 3. 调用nextInt(100)+1(写在for里面) 4. 10次循环的for🡪往数组中放随机数 5. 创建一个集合 6. 遍历数组,判断遍历出来的元素是不是大于等于10,如果是,调用add添加到集合中 |

**public class** CollectionTest08 {  
 **public static void** main(String[] args) {  
 //1.产生10个1-100的随机数,把数组中大于等于10的数字放到一个list集合中，并打印到控制台。  
 //(1)定义长度为10的int数组  
 **int**[] arr = **new int**[10];  
 //(2)创建产生随机数的对象  
 Random r = **new** Random();  
 //(3)产生随机数,并存入数组中  
 **for** (**int** i = 0; i < arr.**length**; i++) {  
 arr[i] = r.nextInt(100) + 1;  
 }  
 //(4)把数组中大于等于10的数字放到一个list集合中，并打印到控制台。  
 //定义List集合  
 ArrayList<Integer> list = **new** ArrayList<>();  
 //遍历arr数组,将>=10的元素存入到list集合中  
 **for** (Integer thisNum : arr) {  
 **if** (thisNum >= 10) {  
 list.add(thisNum);

}  
 }  
 System.**out**.println(**"产生的随机数是:"** + list);  
 }  
}

### 练习十一：泛型方法

1. 编写一个泛型方法，实现任意引用类型数组指定位置元素交换。

**public class** Demo09 {  
 **public static void** main(String[] args) {  
 Integer[] a = {1,2,3,4,5,6};  
 method(a,0,1);  
 }  
 //编写泛型方法  
 **public static** <E> **void** method( E[] e,**int** a,**int** b){  
 //元素互换  
 E temp = e[a];  
 e[a] = e[b];  
 e[b] = temp;  
 **for** (**int** i = 0; i < e.**length**; i++) {  
 System.**out**.println(e[i]);  
 }  
 }  
}

### 练习十二：泛型方法

|  |
| --- |
| 主要考虑的问题:  泛型的方法怎么定义:  修饰符 <E> 返回值 方法名(E[] e)  在main方法中怎么定义这个数组:  Integer[] arr = new Integer [长度] |

1. 编写一个泛型方法，接收一个任意引用类型的数组，并反转数组中的所有元素

**public class** Demo13 {  
 **public static void** main(String[] args) {  
 Integer[] a = {1,2,3,4,5,6};  
 method(a);  
 }  
 //编写泛型方法  
 **public static** <E> **void** method( E[] e){  
 //元素反转  
 **for** (**int** min = 0,max = e.**length** - 1; min < max; min++,max--) {  
 E temp = e[min];  
 e[min] = e[max];  
 e[max] = temp;  
 }  
 **for** (**int** i = 0; i < e.**length**; i++) {  
 System.**out**.println(e[i]);  
 }  
 }  
}