## 第6章 集合类

一、填空题

1． \_Collection\_\_是所有单列集合的父接口，它定义了单列集合（List和Set）通用的一些方法。

2．使用Iterator遍历集合时，首先需要调用\_hasNext()\_\_方法判断是否存在下一个元素，若存在下一个元素，则调用\_\_next()\_\_方法取出该元素。

3．List集合的主要实现类有\_ArrayList\_\_、\_LinkedList\_\_，Set集合的主要实现类有\_\_HashSet\_\_、\_TreeSet\_\_，Map集合的主要实现类有\_\_HashMap\_\_、\_\_TreeMap\_\_。

4．Map接口是一种双列集合，它的每个元素都包含一个键对象\_\_\_Key\_\_和值对象\_\_Value\_\_\_，键和值对象之间存在一种对应关系，称为映射。

5．ArrayList内部封装了一个长度可变的\_\_数组\_\_\_。

二、判断题

1．Set集合时通过键值对的方式来存储对象的。（×）

2．ArrayList集合查询元素的速度很快，但是增删改查效率较低。 （√）

3．Set接口主要有两个实现类，分别是HashSet和TreeSet。（√）

4．Map接口是一种双列集合，它的每个元素都包含一个键对象Key和值对象Value。（√）

5．Lambda表达式只能是一个语句块。（×）

三、选择题

1．以下那些集合可以保存具有映射关系的数据（ B、C）（多选）

A、ArrayList

B、TreeMap

C、HashMap

D、TreeSet

2．下列关于LinkedList类的方法，不是从List接口中继承而来的是（B ）

A、toArray()

B、pop()

C、remove()

D、isEmpty()

3．以下属于Map接口集合常用方法的有（ A、B、C、D）（多选）

A、boolean containsKey(Object key)

B、Collection values()

C、void forEach(BiConsumer action)

D、boolean replace(Object key, Object value)

4．使用Iterator时，判断是否存在下一个元素可以使用以下哪个方法（ D ）

A、next()

B、hash()

C、hasPrevious()

D、hasNext()

5．阅读下面的代码：

public class Example{

    public static void main(String[] args) {

        String[] strs = { "Tom", "Jerry", "Donald" };

        // foreach循环遍历数组

        for (String str : strs) {

            str = "Tuffy";

        }

        System.out.println(strs[0]+ "," + strs[1] + "," + strs[2]);

    }

}

程序的运行结果是（C ）

A. Tom,Jerry

B. Tom,Jerry, Tuffy

C. Tom,Jerry,Donald

D. 以上都不对

四、简答题

1．简述集合List、Set和Map的区别。 List：有序、可重复、可索引；Set：无序、不重复；Map：可重复、存储的是键、值的映射关系。

2．简述为什么ArrayList的增删操作比较慢,查找操作比较快。 由于ArrayList集合的底层是使用一个数组来保存元素,在增加或删除指定位置的元素时,会导致创建新的数组,效率比较低,因此不适合做大量的增删操作。但这种数组的结构允许程序通过索引的方式来访问元素,因此使用ArrayList集合查找元素很便捷。

五、编程题

1. 请按照下列要求编写程序。

（1） 编写一个Student类，包含name和age属性，提供有参构造方法。

（2） 在Student类中，重写toString()方法，输出age和name的值。

（3） 在Student类中，重写hashCode()和equals()方法 a.hashCode()的返回值是name的hash值与age的和。b.equals()判断对象的name和age是否相同，相同则返回true不同返回false。

（4）最后编写一个测试类，创建一个HashSet<Student>对象hs，向hs中添加多个Student对象，假设有两个Student对象相等，输出HashSet，观察是否添加成功。

2. 请按照下列提示编写一个泛型接口以及其实现类。

提示：

（1）创建一个泛型接口Generic<T>，并创建一个抽象方法get(T t)；

（2）创建一个实现类GenericImpl<T>，空实现get(T t)方法。

1、import java.util.\*;

class Student{

private int age;

private String name;

public Student(int age,String name){

this.age=age;

this.name=name;

}

public String toString(){

return age+":"+name;

}

public int hashCode() {

return name.hashCode() + age;

}

public boolean equals(Object obj) {

if(this==obj)

return true;

if(!(obj instanceof Student))

return false;

Student stu=(Student) obj;

return

this.name.equals(stu.name)&&this.age==stu.age;

}

}

class HashSetDemo{

public static void main(String[] args){

HashSet<Student>hs=new HashSet<Student>();

hs.add(new Student(18,"zhangsan"));

hs.add(new Student(20,"lisa"));

hs.add(new Student(20,"lisa"));

System.out.println(hs);

}

}

2、interface Generic<T>{

public abstract void get(T t){}

}

class Geneic<T>implements Generic{

public void get(T t){}

}