《Java 程序设计》实验报告 15

学生姓名：李季鸿　　班级：2021 级计本（3）班　　学号：20213002624

实验地点：9 教 404　　指导教师：张春元

实验日期：2023-06-05　　共 2 学时

实验环境：Win10+JDK1.8+ IntelliJ IDEA 2022.1.1

1. 实验目的

复习数组、类、继承与多态、接口、泛型编程。

2. 实验内容

（1） 编写程序打印输出教材 P31 数组 a 中的最大值。

（2） 复现教材 P135 例 8。

（3） 复现教材 P167 例 6。

（4） 复现教材 P463 上半页例子。

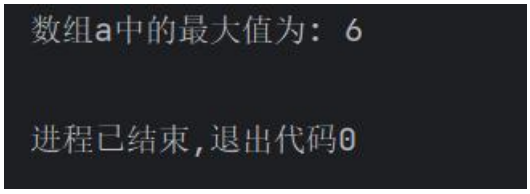
3. 实验过程

报告撰写具体要求：截屏显示或直接写出实验 1 至实验 4 源码和运行结果。

实验内容（1）：

```
package _15_.shiyan1;

/**
* \* Created with IntelliJ IDEA.
* \* @ProjectName: java_study_codes
* \* @FileName: FindMax
* \* @author: li-jihong
* \* Date: 2023-06-05 15:04
*/
public class FindMax {
    public static void main(String[] args) {
        int[][] a = {{1}, {1, 1}, {1, 2, 1}, {1, 3, 3, 1}, {1, 4, 6, 4, 1}};

        int max = Integer.MIN_VALUE;

        for (int[] row : a) {
            for (int value : row) {
                if (value > max) {
                    max = value;
                }
            }
        }

        System.out.println("数组 a 中的最大值为: " + max);
```

```
    }
}
```



实验内容（2）：

```
package _15_.shiyan2;

/**
* \* Created with IntelliJ IDEA.
* \* @ProjectName: java_study_codes
* \* @FileName: Main
* \* @author: li-jihong
* \* Date: 2023-06-05 15:09
*/
class Student {
    int number;
    String name;

    Student() {
    }

    Student(int number, String name) {
        this.number = number;
        this.name = name;
        System.out.println("我的名字是:" + name + "  学号是:" + number);
    }
}

class UniverStudent extends Student {
    boolean  婚否;

    UniverStudent(int number, String name, boolean b) {
        super(number, name);
        婚否  = b;
        System.out.println("婚否=" +  婚否);
    }
}

public class Main {
    public static void main(String[] args) {
```

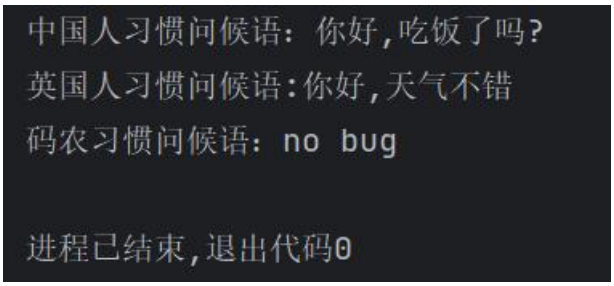
```
        UniverStudent zhang = new UniverStudent(9901, "何晓林", false);
    }
}
```

```
我的名字是:何晓林  学号是:9901
婚否=false

进程已结束,退出代码0
```

实验内容（3）：

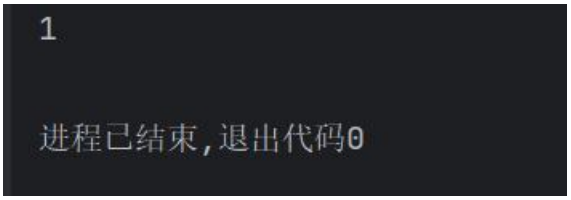
```
package _15_.shiyan3;

interface SpeakHello {
    void speakHello();
}

class Chinese implements SpeakHello {
    public void speakHello() {
        System.out.println("中国人习惯问候语：你好,吃饭了吗? ");
    }
}

class English implements SpeakHello {
    public void speakHello() {
        System.out.println("英国人习惯问候语:你好,天气不错  ");
    }
}

class KindHello {
    public void lookHello(SpeakHello hello) { //接口类型参数
        hello.speakHello();                          //接口回调
    }
}

public class Main {
    public static void main(String[] args) {
        KindHello a = new KindHello();
        Chinese ccc = new Chinese();
        a.lookHello(ccc);
        a.lookHello(new English());
        a.lookHello(() -> System.out.println("码农习惯问候语：no bug"));//向参数
传递 Lambda 表达式的值
    }
```

}



实验内容（4）：

```
package _15_.shiyan4;

/**
* \* Created with IntelliJ IDEA.
* \* @ProjectName: java_study_codes
* \* @FileName: Dog
* \* @author: li-jihong
* \* Date: 2023-06-05 15:14
*/
public class Dog implements Comparable<Dog> {
    int weight;

    public static void main(String[] args) {
        Dog dog1 = new Dog();
        Dog dog2 = new Dog();
        dog1.weight = 20;
        dog1.weight = 25;
        System.out.println(dog1.comparaTo(dog2));
    }

    @Override
    public int comparaTo(Dog m) {
        if (weight > m.weight) {
            return 1;
        } else if (weight == m.weight) {
            return 0;
        } else {
            return -1;
        }
    }
}
```



4. 实验总结

写出实验中的心得体会（对第 15 章理论课重点简述）。

一、泛型类定义

可以使用“class 名称<泛型列表>”定义一个类，为了和普通的类有所区别，这样定义的类称作泛型类，如：

class People<E,V> {……}

其中：People 是泛型类的名称，E 和 V 是该类用到的泛型。

泛型表示任意的复合数据类型（即类和接口），但不表示基本数据类型，使用一个合理的标识符都可以。

在一个泛型类中，泛型可以作为类的成员变量的类型、方法的形参类型和局部变量的类型。

二、使用泛型类声明和创建对象

泛型类声明和创建对象时，类名后多了一对“<>”，而且必须要用具体的类型替换“<>”中的泛型。例如：

People<LocalTime> zhang;

zhang =new People<LocalTime>();

三、实现泛型接口

一个非泛型类实现泛型接口时，必须指定泛型接口中泛型的具体类型。比如，java.lang 包中的泛型接口

public interface Comparable<T> {

public int compareTo(T m);

}

教材 P463 Dog 类实现 Comparable<T>接口时，将泛型

T 指定为具体的 Dog 类型

四、链表

链表是由若干个称作结点的对象组成的一种数据结构，每结结点含有一个数据和下一个结点的引用。

五、Stack<E>泛型类

堆栈是一种“先进后出”的数据结构，只能在一端进行输入或输出数据的操作。

Stack<E>创建一个堆栈对象，堆栈对象常用方法：

public E push(E item); 实现压栈操作

public E pop(); 实现弹栈操作。

public boolean empty(); 判断堆栈是否还有数据。

public E peek(); 获取堆栈顶端的数据，但不删除该数据。

public int search(Object data);获取数据在堆栈中的位置。

六、HashMap<K,V>泛型类

HashMap<K,V>表示散列表数据结构，习惯上称 HashMap<K,V>为散列映射类，K 为键，V 为值，用来存储“键/值”对数据。

例如:

HashMap<String,Student> hashtable = HashSet<String,Student>();

相关方法：

public V put(K key,V value)将键/值对数据存放到散列映射中，该方法同时返回键所对应的值。

七、TreeSet<E>泛型类

TreeSet<E>创建的对象称作树集，树集采用二叉树结构存储数据，树结点中的数据会按存放的数据的“大小”

顺序一层层地依次排列，在同一层中的结点从左到右按字典序从小到大递增排列，下一层都比上一层的要小。

TreeSet<E>适用于数据排序。

例如：

```
TreeSet<String> mytree=new TreeSet<String>();
mytree.add(“boy”); //为树集添加结点
```

八、TreeMap<K,V>泛型类

TreeMap<K,V>类实现了 Map<K,V>接口，称 TreeMap<K,V>对象为树映射。树映射使用 public V put(K key,V value);方法添加结点。

和树集不同，树映射保证结点按关键字升序排列。

九、HashSet<E>泛型类创建的对象称为集合，如

```
HashSet<String> set= HashSet<String>();
```

那么 set 就是一个可以存储 String 类型数据的集合，set 可以调用 add(String s)方法将 String 类型的数据添加到集合中，添加到集合中的数据称为集合的元素。集合不允许有相同的元素，也就是说，如果 b 已经是集合中的元素，那么再执行 set.add(b)操作是无效的。集合对象的初始容量是 16 个字节，装载因子是 0.75。也就是说，如果集合添加的元素超过总容量的 75%时，集合的容量将增加一倍。

集合对象调用 boolean addAll(HashSet set)方法可以与参数指定的集合求并运算，使得当前集合成为两个集合的并。

集合对象调用 boolean retainAll (HashSet set)方法可以与参数指定的集合求交运算，使得当前集合成为两个集合的交。

集合对象调用 boolean removeAll (HashSet set)方法可以与参数指定的集合求差运算，使得当前集合成为两个集合的差。