

云南大学数学与统计学院

上机实践报告

课程名称：数据结构与算法实验	年级：2013	上机实践成绩：
指导教师：陆正福	姓名：金洋	
上机实践名称：Java 面向对象编程实验	学号：20131910023	上机实践日期： 3.24
上机实践编号：No. 2	组号：	上机实践时间： 18:00

一、实验目的

- 1.熟悉 Java 面向对象编程，为数据结构与算法的学习奠定实验基础
- 2.熟悉教材第二章的代码片段

二、实验内容

1. 熟悉 Java 面向对象技术：封装、继承、多态、设计模式、程序代码的组织结构等
2. 调试主讲教材第二章的 java 程序

三、实验平台

个人计算机; Oracle/Sun Java 7 SE 或 EE

四、实验记录与实验结果分析

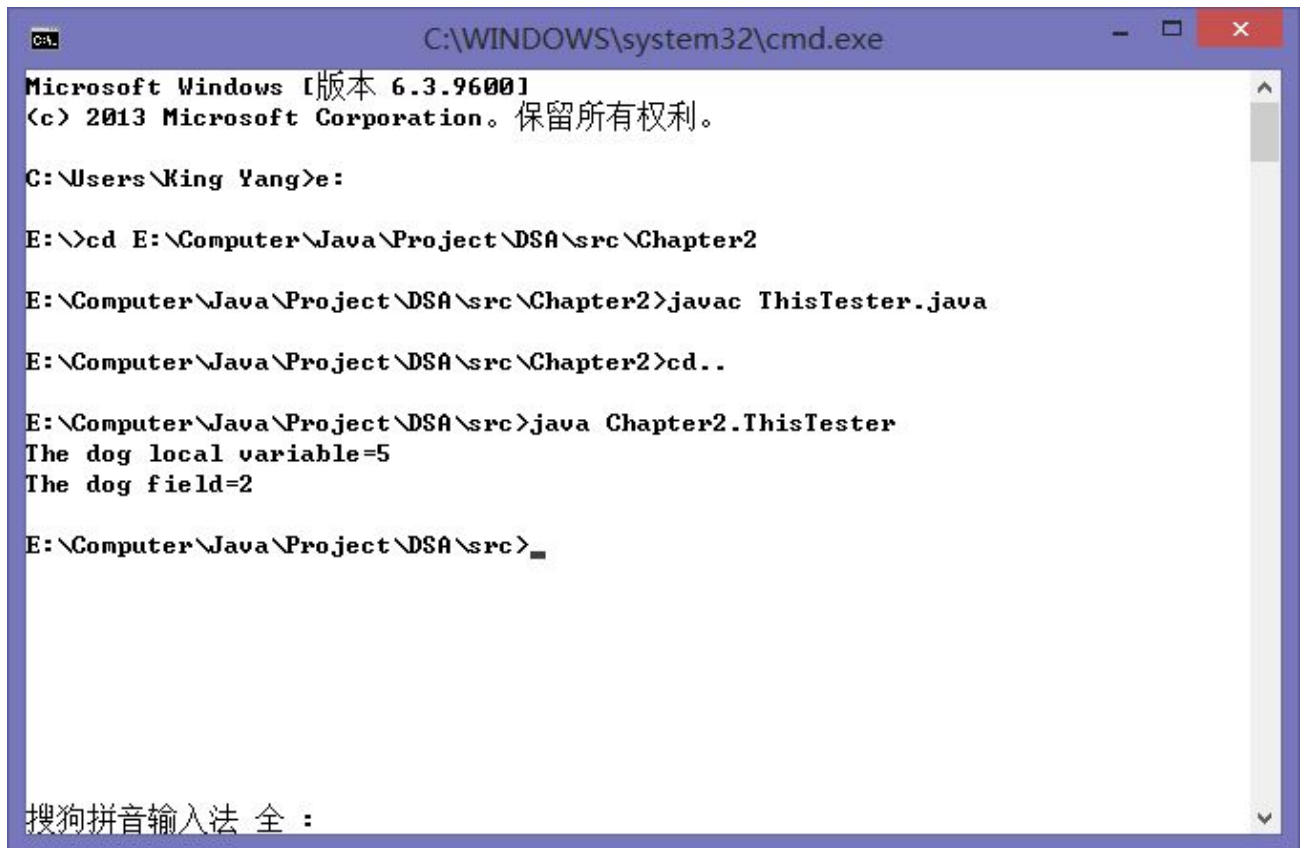
（注意记录实验中遇到的问题。实验报告的评分依据之一是实验记录的细致程度、实验过程的真实性、实验结果的解释和分析。如果涉及实验结果截屏，应选择白底黑字。）

1.多个类组成一个完整的程序时，考虑将这几个类的源代码放入同一个包中，则在每个类的源代码的开头添一句” package 包名 A;”

例如在文件夹..\Chapter2 中放着 ThisTester.java,该源程序首句为 package Chapter2;

编译：则在 dos 环境下，先到达..\Chapter2 目录下，使用 javac ThisTester.java 进行编译；(和不带 package 的程序运行方式相同)

运行：cd.. 退回到包文件的上一层目录，再键入:java Chapter2.ThisTester(和不带 package 的程序运行相同)



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [版本 6.3.9600]
(c) 2013 Microsoft Corporation。保留所有权利。

C:\Users\King Yang>e:

E:\>cd E:\Computer\Java\Project\DSA\src\Chapter2

E:\Computer\Java\Project\DSA\src\Chapter2>javac ThisTester.java

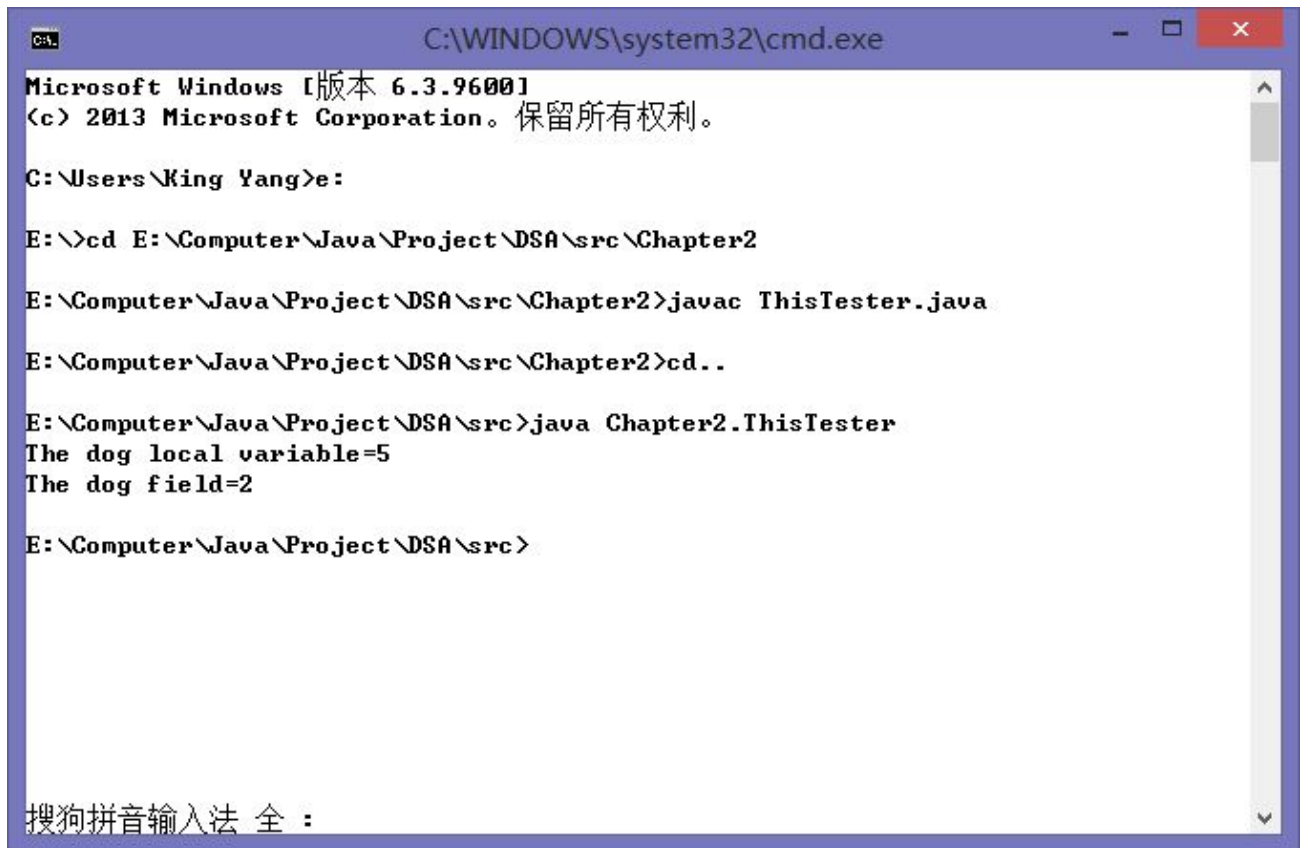
E:\Computer\Java\Project\DSA\src\Chapter2>cd..

E:\Computer\Java\Project\DSA\src>java Chapter2.ThisTester
The dog local variable=5
The dog field=2

E:\Computer\Java\Project\DSA\src>
```

2. //关键字 this 的使用方法

```
public class ThisTester {
    public int dog=2;//instance variable;
    public void clobber(){
        int dog=5;//a different dog!
        System.out.println("The dog local variable="+dog);
        System.out.println("The dog field="+this.dog);
    }
    public static void main(String[] args) {
        ThisTester t=new ThisTester();
        t.clobber();
    }
}
```



```

C:\WINDOWS\system32\cmd.exe
Microsoft Windows [版本 6.3.9600]
(c) 2013 Microsoft Corporation。保留所有权利。

C:\Users\King Yang>e:

E:\>cd E:\Computer\Java\Project\DSA\src\Chapter2

E:\Computer\Java\Project\DSA\src\Chapter2>javac ThisTester.java

E:\Computer\Java\Project\DSA\src\Chapter2>cd..

E:\Computer\Java\Project\DSA\src>java Chapter2.ThisTester
The dog local variable=5
The dog field=2

E:\Computer\Java\Project\DSA\src>

```

搜狗拼音输入法 全 :

3. 级数

```

//A class for numeric progression
public class Progression {
    //First value of the progression
    protected long first;
    //Current value of the progression
    protected long cur;
    //Default constructor
    ///构造函数的特点：①与类名同名 ②无返回值
    Progression() {
        cur=first=0;
    }

    /*Reset the progression to the first value
    * @return first value
    */
    protected long firstValue() {
        cur=first;
        return cur;
    }

    /*Advances the progression to the next value
    * @return next value of the progression
    */
    protected long nextValue() {
        return ++cur;//default next value
    }

    /*Print the first n value of the progression

```

```

        * @param n number of values to print
        */
    public void printProgression(int n){
        System.out.print(firstValue());
        for(int i=2;i<=n;i++){
            System.out.print(" "+nextValue());
            System.out.println();//ends the line
        }
    }

//Arithmetic progression.
public class ArithProgression extends Progression{
    protected long inc;//Increment
    //Inherits variables first and cur;

    //Default constructor setting a unit increment
    ArithProgression(){
        this(1);//this 关键字可以出现在类的实例方法中，代表使用该方法的当前对象；
        //往下找同名的方法，1 为参数
    }
    //Parametric constructor providing the increment
    ArithProgression(long increment){
        inc=increment;
    }
    /*Advances the progression by adding the increment to the current value
    * @return next value of the progression
    */
    protected long nextValue(){
        cur+=inc;
        return cur;
    }
}

//Geometric Progression
public class GeomProgression extends Progression{
    protected long base;//Base
    //Inherits variables first and cur

    //Default constructor setting base 2
    GeomProgression(){
        this(2);
    }
    /*Parametric constructor providing the base
    * @param b base of the progression
    */
    GeomProgression(long b){
        base=b;
        first=1;
        cur=first;
    }
    /*Advances the progression by multiplying the base with the currnt value

```

```

        * return next value of the progress
        */
protected long nextValue() {
    cur*=base;
    return cur;
}
//Inherits methods firstValue() and printProgression(int)
}

//Fibonacci progression

public class FibonacciProgression extends Progression {
    //Previous value
    long prev;
    //Inherits variables first and cur

    //Default constructor setting 0 and 1 as the first two values
    FibonacciProgression() {
        this(0, 1);
    }
    /*Parametric constructor providing the first and second values
    * @param value1 first value.
    * @param value2 second value.
    */
    public FibonacciProgression(long value1, long value2) {
        first=value1;
        prev=value2-value1;
    }
    protected long nextValue() {
        long temp=prev;
        prev=cur;
        cur+=temp;
        return cur;
    }
    //Inherits methods firstValue() and printProgrss(int)
}

public class TestProgression {
    public static void main(String[] args) {
        Progression prog;

        //test ArithProgression
        System.out.println("Arithmetic progression with default increment:");
        prog=new ArithProgression();
        prog.printProgression(10);
        System.out.println("Arithmetic progression with increment5:");
        prog=new ArithProgression(5);
        prog.printProgression(10);

        //test GeomProgression
        System.out.println("Geometric progression with default base:");
        prog=new GeomProgression();
    }
}

```

```

        prog.printProgression(10);
        System.out.println("Geometric progression with base 3:");
        prog=new GeomProgression(3);
        prog.printProgression(10);

        //test FibonacciProgression
        System.out.println("Fibonacci progression with default start values:");
        prog=new FibonacciProgression();
        prog.printProgression(10);
        System.out.println("Fibonacci progression with default start values 4 and 6:");
        prog=new FibonacciProgression(4,6);
        prog.printProgression(10);

    }
}

```

```

return cur;
^
3 错误
E:\Computer\Java\Project\DSA\src\Chapter2>javac TestProgression.java
E:\Computer\Java\Project\DSA\src\Chapter2>java TestProgression
Arithmetic progression with default increment:
0 1 2 3 4 5 6 7 8 9
Arithmetic progression with increment5:
0 5 10 15 20 25 30 35 40 45
Geometric progression with default base:
1 2 4 8 16 32 64 128 256 512
Geometric progression with base 3:
1 3 9 27 81 243 729 2187 6561 19683
Fibonacci progression with default start values:
0 1 1 2 3 5 8 13 21 34
Fibonacci progression with default start values 4 and 6:
4 6 10 16 26 42 68 110 178 288
E:\Computer\Java\Project\DSA\src\Chapter2>

```

4. 接口

```

public interface Sellable { //接口的方法体没有体部
    public String description();
    public int listPrice();
    public int lowestPrice();
}

public class Photograph implements Sellable { //类实现了接口
    private String descript;

```

```

    private int price;
    private boolean color;

    public Photograph(String desc, int p, boolean c) {
        descript=desc;
        price=p;
        color=c;
    }

    public String description() {return descript;}//有体部，实现了接口
    public int listPrice() {return price;}
    public int lowestPrice() {return price/2;}
    public boolean isColor() {return color;}
}

public interface Transportable {
    public int weight();
    public boolean isHazardous();
}

public class BoxedItem implements Sellable, Transportable{//多实现
    private String descript;
    private int price, weight, height=0, width=0, depth=0;
    private boolean haz;
    //Constructor
    public BoxedItem(String desc, int p, int w, boolean h) {
        descript=desc;
        price=p;
        weight=w;
        haz=h;
    }
    public String description() {return descript;}
    public int listPrice() {return price;}
    public int lowestPrice() {return price/2;}
    public int weight() {return weight;}
    public boolean isHazardous() {return haz;}
    public int insuredValue() {return price*2;}
    public void setBox(int h, int w, int d) {
        height=h;
        width=w;
        depth=d;
    }
}

```

5. 泛型

```

public interface Person {
    public boolean equalTo(Person other);
    public String getName();
    public int getAge();
}

```

```

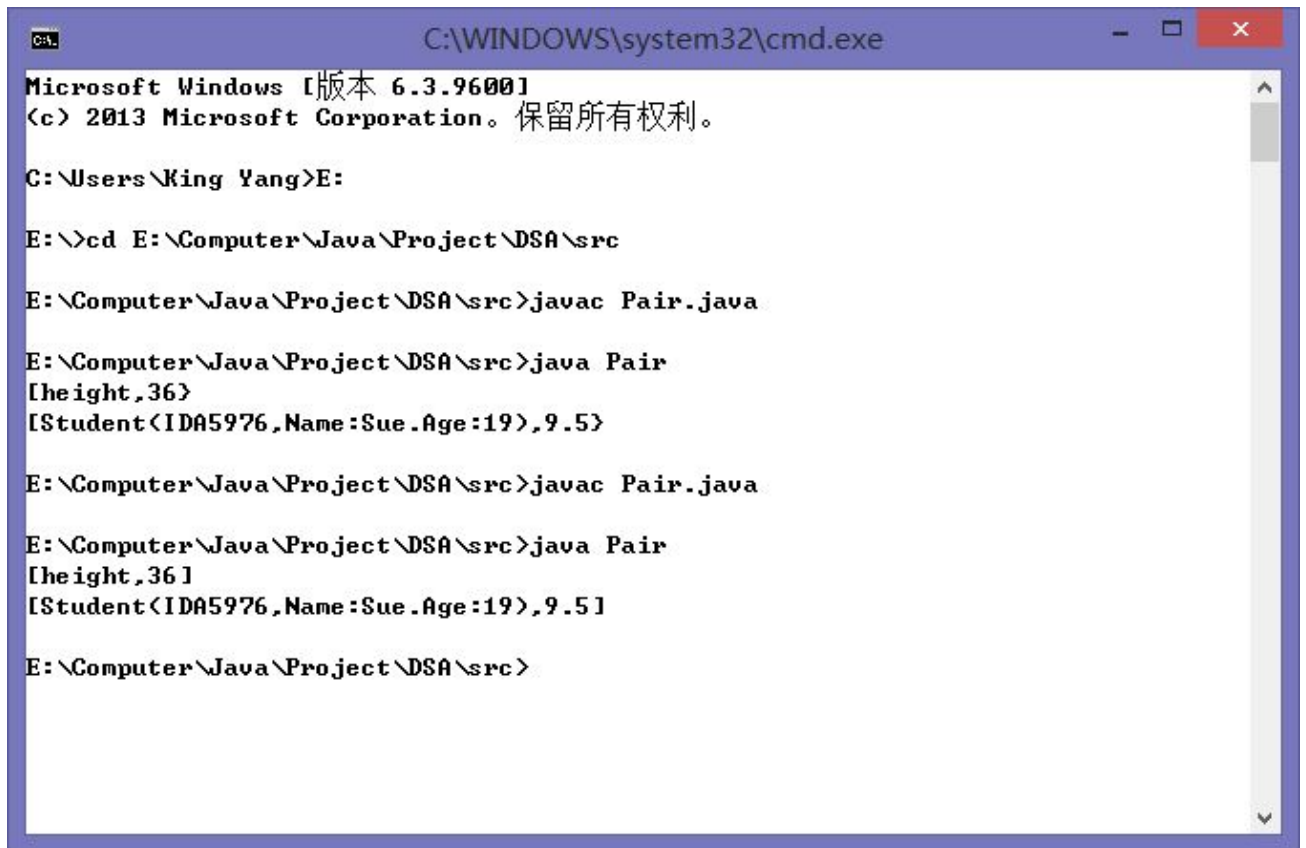
public class Student implements Person {
    String id;
    String name;
    int age;
    public Student(String i,String n,int a) {
        id =i;
        name=n;
        age=a;
    }
    protected int studyHours() {return age/2;}
    public String getID() {return id;}
    public String getName() {return name;}
    public int getAge() {return age;}
    public boolean equalTo(Person other) {
        Student otherStudent=(Student) other;
        return (id.equals(otherStudent.getID()));
    }
    public String toString() {

        return "Student (ID"+id+", Name:"+name+". Age:"+age+")";
    }

}

public class Pair<K,V> { //键值对 大写字母是形式类型上的参数
    K key;//自变量
    V value;//因变量
    public void set(K k,V v) {
        key=k;
        value=v;
    }
    public K getKey() {return key;    }
    public V getValue() {return value;}
    public String toString() {
        return "["+getKey()+","+getValue()+"]";
    }
    public static void main(String[] args) {
        Pair<String,Integer> pair1 =new Pair<String,Integer>();
        pair1.set(new String("height"), new Integer(36));
        System.out.println(pair1);
        Pair<Student,Double> pair2=new Pair<Student,Double>();
        pair2.set(new Student("A5976","Sue",19),new Double(9.5));
        System.out.println(pair2);
    }
}

```

```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [版本 6.3.9600]
(c) 2013 Microsoft Corporation. 保留所有权利。

C:\Users\King Yang>E:

E:\>cd E:\Computer\Java\Project\DSA\src

E:\Computer\Java\Project\DSA\src>javac Pair.java

E:\Computer\Java\Project\DSA\src>java Pair
[height,36]
[Student<IDA5976,Name:Sue.Age:19>,9.5]

E:\Computer\Java\Project\DSA\src>javac Pair.java

E:\Computer\Java\Project\DSA\src>java Pair
[height,36]
[Student<IDA5976,Name:Sue.Age:19>,9.5]

E:\Computer\Java\Project\DSA\src>
```

五、实验体会

1. 多个类组成一个完整的程序时，考虑将这几个类的源代码放入同一个包中，在 dos 环境下，编译的方式与不带 package 的程序相同；运行时，退回到包的上一级目录，键入 java 包名. 程序名，（运行方式同不带 package 的程序有区别）；

2. this 是 Java 的一个关键字，可以出现在实例方法和构造方法中，但不可以出现在类方法中；

①. 在构造方法中使用 this

this 关键字可以出现在类的构造方法中，代表使用该构造方法所创建的对象；

②. 在实例方法中使用 this

this 关键字可以出现在类的实例方法中，代表使用该方法的当前对象；

3. 构造函数的特点：①与类名同名 ②无返回值

4. Java 不支持多继承性，即一个类只能有一个父类。单继承性使得 Java 简单，易于管理程序。为了克服单继承的缺点，Java 使用了接口，一个类可以实现多个接口；

5. 泛型类声明

可以使用“class 名称<泛型列表>”声明一个类，为了和普通的类有所区别，这样声明的类称作泛型类，如：class A<E >

其中 A 是泛型类的名称，E 是其中的泛型，也就是说我们并没有指定 E 是何种类型的数据，它可以是任何对象或接口，但不能是基本类型数据；

6. 使用泛型类声明对象

使用泛型类声明对象时，必须要指定类中使用的泛型的具体实际类型；

六、参考文献

1. 主讲课英文教材 **Goodrich, Tamassia: *Data Structures and Algorithms in Java*, 5th Edition International Student Version chapter 2**

2.邱仲潘, 朱诗兵, 朱小谷.Java 程序设计教程.红旗出版社,2005.3