沈阳工学院 实 验 报 告

实验项目名称：JAVA 面向对象编程

（所属课程：JAVA 语言程序设计）

院　　系：经济管理学院　专业班级：15 信息管理专升本　姓　名：马高飞、张廷立、崔颢　学　号：15073730121、15073730120、15073730138

实验日期：　实验地点：11 号楼 206　合作者：　指导教师：张莉

本实验项目成绩：　**教师签字：**　**日期：**

一、实验目的

1、理解 Java 语言是如何体现面向对象编程基本思想，

2、掌握类的封装方法，以及如何创建类和对象，

3、掌握成员变量和成员方法的特性。

4、掌握 OOP 方式进行程序设计的方法。

5、掌握类的继承性和多态性的作用。

二、实验条件

Windows XP，JDK1.6 与 Eclipse

三、实验内容

1、创建对象和使用对象的方法程序。

2、不同成员变量修饰方法的程序。

3、不同成员方法修饰的程序。

4、体现类多态性（成员方法重载，构造方法重载）的程序。

四、实验步骤

1．实例变量/方法与静态变量/方法（或称类变量/类方法）

静态变量/方法前面加 static 修饰符，访问时通过(类名. 变量/方法名)或（对象名. 变量/方法名）。实例变量/方法前面不加 static 修饰符，访问时通过（对象名. 变量/方法名）。

通过两个类 StaticDemo、LX2_1 说明静态变量/方法与实例变量/方法的区别。

编写类文件 LX2_1.java，程序源代码如下。

```
class StaticDemo {
static int x;
int y;
public static int getX() {
return x;
}
public static void setX(int newX) {
x = newX;
}
public int getY() {
return y;
}
```

```
public void setY(int newY) {

y = newY;

}

}

public class LX2_1 {

public static void main(String[] args) {

System.out.println("静态变量 x="+StaticDemo.getX());

System.out.println("实例变量 y="+StaticDemo.getY()); // 非法，编译将出错

StaticDemo a= new StaticDemo();

StaticDemo b= new StaticDemo();

a.setX(1);

a.setY(2);

b.setX(3);

b.setY(4);

System.out.println("静态变量 a.x="+a.getX());

System.out.println("实例变量 a.y="+a.getY());

System.out.println("静态变量 b.x="+b.getX());

System.out.println("实例变量 b.y="+b.getY());

}
```

}

（1）编译该源代码，观察结果，分析并给出错误的原因。

（2）将程序中的出错语句使用注释符//隐藏起来，重新编译并运行，给出结果。

2、类成员的访问控制，使用修饰符。

有时需要公开一些变量和方法，有时需要禁止其他对象使用变量和方法，这时可以使用修饰符来实现这个目的。

常用的修饰符如下。[public] [private] [protected] [package] [static] [final] [transient] [volatile]不同修饰符的访问控制权限如表 2.1 所示。

修饰符	类	子类	包	所有类和包
public	√	√	√	√
private	√			
protected	√	√*	√	
package	√		√	

通过两个类 AccessDemo、LX2_2 说明 public、private 与无修饰符的访问控制。

编写类文件 LX2_2.java，程序源代码如下。

```
class AccessDemo {
public int x;
private int y;
int z;
}
public class LX2_2 {
```

```
    public static void main(String[] args) {
    AccessDemo a= new AccessDemo();
    a.x=1;
    a.y=2;
    a.z=3;
    System.out.println("public 成员变量 a.x="+a.x);
    System.out.println("private 成员变量 a.y="+a.y);
    System.out.println("无修饰符成员变量 a.z="+a.z);
    }
}
```

(1)编译该源代码，观察结果，分析并给出错误的原因。

(2)修改源程序，添加 private 成员变量的 public get/set 方法，使程序可以运行。给出核心代码和实验结果。

3、方法中参数传递的练习

在 Java 中，方法中的参数传递可以分为传值调用或传递引用（传地址）调用方式。

传值调用即传递的参数是基本数据类型，调用方法时在方法中将不能改变参数的值，相当于传递副本。

传递引用传递的参数为对象或者数组，调用方法时在方法中能改变参数的值，相当于传递本身。

编写程序文件 LX3_3.java，实现传值和传引用，比较二者的区别，程序源代码如下。

```
class Paratran{
int x=10, y=10;
}
class ChangePara{
void change1(int passX, int passY) {
System.out.println("初始时 x="+ passX +", y="+ passY);
passX =passX*passX;
passY =passY*passY;
System.out.println("方法调用中 x="+ passx +", y="+ passY);
}
void change2(Paratran a) {
System.out.println("初始时 x="+ a.x +", y="+ a.y);
a.x =a.x*a.x;
a.y =a.y*a.y;
System.out.println("方法调用中 x="+ a.x +", y="+ a.y);
}
}
Public class LX3_3 {
public static void main(String[] args) {
```

```
Paratran p=new Paratran();
ChangePara c= new ChangePara();
c.change1(p.x,p.y); //传值调用，传递的是基本类型 int
c.change2(p);  //传地址调用，传递的是对象
System.out.println("方法调用后 x="+p.x+", y="+p.y);
}
}
```

（1）使用//注释符号将 c.change2(p);隐藏，编译运行该源代码，给出实验结果。

（2）恢复 c.change2(p)的隐藏，使用//注释符号将 c.change1(p.x,p.y);隐藏，编译运行该源代码，给出实验结果。

4、类的多态（借助方法的重载实现）

类的继承发生在多个类之间，而类的多态只发生在同一个类上。

在一个类中，可以定义多个同名的方法，只要确定它们的参数个数和类型不同，这称之为方法的重载，表现为类的多态。

在OOP 中，当程序要实现多个相近的功能时，就给相应的方法起一个共同的名字，用不同的参数代表不同的功能。这样，在使用方法时不论传递什么参数，只要能被程序识别就可以得到确定的结果。

类的多态性体现在方法的重载（overload）上，包括成员方法和构造方法的重载。

（1）成员方法的重载

编写 Addition 类，该类中应包含一组实现两数相加运算的重载方法。

实现加法运算的方法，应接受两个参数（即加数和被加数），方法将两个参数进行加法

运算后，返回相加结果。考虑可能针对不同的数据类型进行计算，重载一组方法，包括整型、长整型、浮点型、双精度浮点型、还有字符串。

在 main 方法中创建 Addition 类的实例，分别调用重载方法测试其效果。

```
package yan.cong;
 class Addition{
     public Addition(){
     }
   void add(int a,int b){
     int c = a+b;
     System.out.println("a+b="+c);
}
   void add(long a,long b){
     long c = a+b;
     System.out.println("a+b="+c);
   }
   void add(float a,float b){
     float c = a+b;
     System.out.println("a+b="+c);
   }
   void add(double a,double b){
        double c = a+b;
        System.out.println("a+b="+c);
```

```
  }
  void add(String a,String b){
      String c = a+b;
      System.out.println("a+b="+c);
  }
}
public class b01
{
    public static void main(String args[]){
        Addition m = new Addition();
        Addition n = new Addition();
        m.add(5,6);
        n.add("hello","world");
    }
}
```

足以看出sort成员方法实现了重载，给出三个sort方法实现的核心代码。

（2）构造方法的重载

构造方法的名称和类同名，没有返回类型。尽管构造方法看起来和一般的成员方法没有差别，但它不是方法，也不是类的成员。因此，构造方法不能直接调用，只能由new操作符调用。构造方法对于类是十分重要的，对象的初始化任务要靠构造方法来完成。

重载构造方法的目的是提供多种初始化对象的能力，使程序员可以根据实际需要选

用合适的构造方法来初始化对象。

编写构造方法RunDemo 的重载程序文件LX3_8，源代码如下。

```
class RunDemo {
private String userName, password;
RunDemo() {
System.out.println("全部为空!");
}
RunDemo(String name) {
userName=name;
}
RunDemo(String name, String pwd) {
this(name);
password=pwd;
check();
}
void check() {
String s=null;
if (userName!=null)
s="用户名："+userName;
else
s="用户名不能为空！";
if (password!="12345678")
```

```
s=s+" 口令无效！";
else
s=s+" 口令：********";
System.out.println(s);
}
}
public class LX3_8 {
public static void main(String[] args) {
new RunDemo();
new RunDemo("刘新宇");
new RunDemo(null,"邵丽萍");
new RunDemo("张驰","12345678");
}
}
```

编译并运行该程序，体会构造方法的重载方式，给出实验结果。

5.构造方法练习

(1)定义一个“点”（Point）类用来表示三维空间中的点（有三个坐标）。要求如下：

1. 可以生成具有特定坐标的点对象。
2. 提供可以设置三个坐标的方法。
3. 提供可以计算该“点”距另外点距离平方的方法。
4. 编写程序验证上述三条。

(2) 编写 Java 程序，模拟简单的计算器。

定义名为 Number 的类，其中有两个整型数据成员 n1 和 n2，应声明为私有。编写构造方法，赋予 n1 和 n2 初始值，再为该类定义加(addition)、减(subtration)、乘（ multiplication ）、除（ division ）等公有成员方法，分别对两个成员变量执行加、减、乘、除的运算。在 main 方法中创建 Number 类的对象，调用各个方法，并显示计算结果。给出核心代码。

6.package 验证练习

将 LX2_1.java 中的两个类分别放到不同的包中，体会不同的权限，以及 import 语句的使用。

```
class StaticDemo {
static int x;
int y;
public static int getX() {
return x;
}
public static void setX(int newX) {
x = newX;
}
public int getY() {
return y;
```

```
}

public void setY(int newY) {

y = newY;

}

}

public class LX2_1 {

public static void main(String[] args) {

System.out.println("静态变量 x="+StaticDemo.getX());

System.out.println("实例变量 y="+StaticDemo.getY()); // 非法，编译将出错

StaticDemo a= new StaticDemo();

StaticDemo b= new StaticDemo();

a.setX(1);

a.setY(2);

b.setX(3);

b.setY(4);

System.out.println("静态变量 a.x="+a.getX());

System.out.println("实例变量 a.y="+a.getY());

System.out.println("静态变量 b.x="+b.getX());

System.out.println("实例变量 b.y="+b.getY());
```

```
}
}
```

7.分析 Leaf.java，画出内存分析图。

```
public class Leaf{
    int i = 0;
    Leaf(int i) {
     this.i = i;
    }
    Leaf increament(){
        i++;
        return this;
    }
    void print(){
     System.out.println("i = "+i);
    }
    public static void main(String[] args){
        Leaf leaf = new Leaf(100);
        leaf.increament().increament().print();
    }
```

```
    }
```

五、实验结果

1、（1）错误原因：静态变量是类固有的，可以直接引用，其他成员变量仅仅被声明，生成实例对象后才存在，才可以被引用。

（2）1234

静态变量a.x=3

实例变量a.y=4

静态变量b.x=3

实例变量b.y=4

2、

```
public class AccessDemo {
    public int x;
        int y;
        int z;
        private void x()
          {
            System.out.println("private x");
          }
      private void y()
          {
            System.out.println("private y");
          }
}
```

结果：public成员变量a.x=1

private成员变量a.y=2

无修饰符成员变量 a.z=3

3、 (1) 结果：初始时 x=10, y=10

方法调用中 x=100, y=100

方法调用后 x=10, y=10

(2) 结果：初始时 x=10, y=10

方法调用中 x=100, y=100

方法调用后 x=100, y=100

4、 (1) 结果：a+b=11

a+b=helloworld

(2) 结果：全部为空!

用户名不能为空！ 口令无效!

用户名：张驰 口令：********

5、 (1)

```
public class Point {
        private double x;
        private double y;
        private double z;
        public Point(int x, int y, int z) {
            }
       public void point() {
        }
```

```
public void point(double x, double y, double z) {
 this.x = x;
 this.y = y;
 this.z = z;
}
public double distance() {
 return x * x + y * y + z * z;
}
public double getX() {
 return x;
}
public void setX(double x) {
 this.x = x;
}
public double getY() {
 return y;
}
public void setY(double y) {
 this.y = y;
}
public double getZ() {
 return z;
```

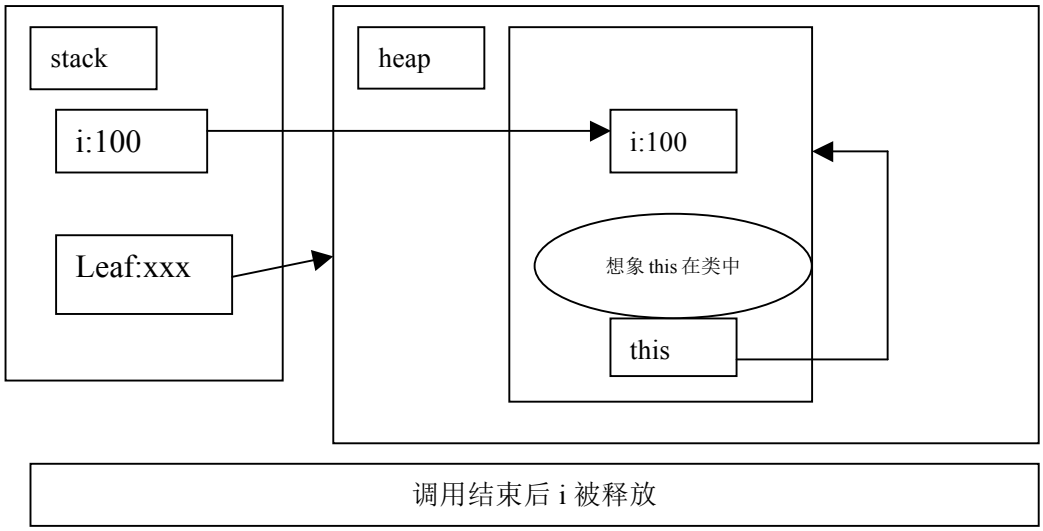
```
        }
        public void setZ(double z) {
          this.z = z;
        }
    }
    public class Test {
      public static void main(String[] args) {
        Point point = new Point(1,2,10);
          Point point1 = new Point(1,2,10);
        point.setX(1);
        point.setY(2);
        point.setZ(10);
        System.out.println(point.distance());
        System.out.println(point1.distance());
       }
    }
```

(2)
```
    public class Number {
    class Number1
    {
    private int n1;
    private int n2;
    Number1(int n1,int n2){
```

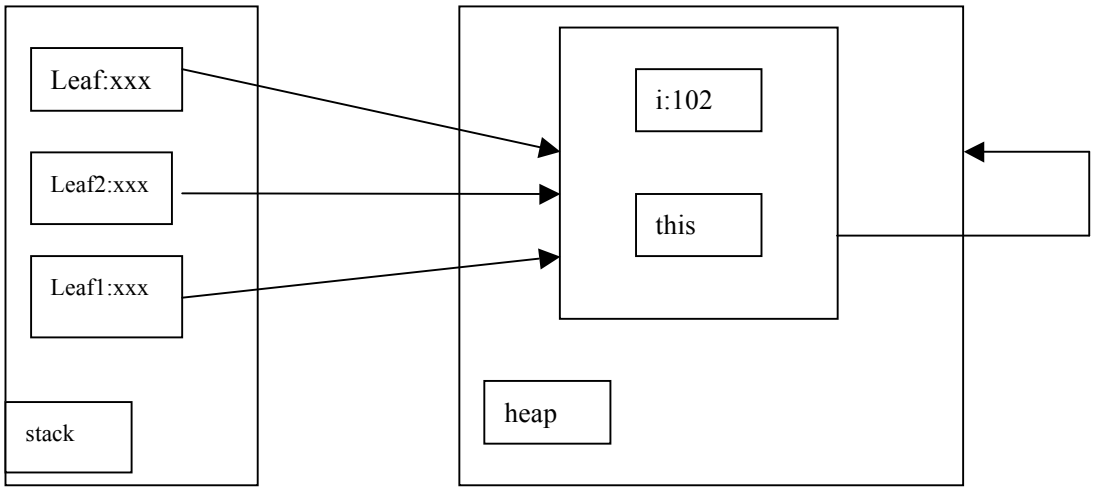
```
    this.n1=n1;
    this.n2=n2;
    }
    public void addition(){
    int equal=n1+n2;
    System.out.println("n1+n2"+equal);
    }
    public void subtration(){
    int equal=n1-n2;
    System.out.println("n1-n2"+equal);
    }
    public void multiplication(){
    int equal=n1*n2;
    System.out.println("n1*n2"+equal);
    }
    public void division(){
    int equal=n1/n2;
    System.out.println("n1/n2"+equal);
    }
}
    public static void main(String args[]){
        Number num = new Number();
```

```
        num.division();
    }
    private void division() {
    }
    }
```

7、Leaf leaf = new Leaf(100);



leaf.increament().increament().print()



This 是当前对象的引用

六、讨论

七、参考文献