第六周实验　继承多态与接口

学号：2015111948　　姓名：王秋锋　　班级：计算机-03 班

实验目的：

- 继承与多态
- 接口的定义、接口的实现

一、类继承相关知识强化

1. 执行以下程序，给出执行 Bbb 的结果，回答问题。

```
class Aaa{
   int i;
   Aaa(int a){ i = a;}
}

classBbb extends Aaa{
   intj,k;
   Bbb(inti){
        super(i);
        j = this.i;
        k = super.i;
   }

   public static void main(String[] args){
      Bbb b = new Bbb(18);
      System.out.println(b.j +"\t" + b.k);
   }
}
```

**************回答问题***********************

（1）执行 Bbb 的结果:

```
18      18
```

（2）this.i 与 super.i 的含义（即它们都对应哪个类中定义的变量）：

This.i 是从类 Aaa 继承下来的类 Bbb 的变量 i,super.i 是类 Aaa 本身的变量 i

2．接上面例子，将类 Bbb 改写为以下代码，回答问题。

```
classBbb extends Aaa{
inti= -1,j= -1,k= -1;   //比上面例子增加一个实例变量 i;
Bbb(inti){
super(i);
j = this.i;        //本语句含义是：
      k = super.i;     //本语句含义是：
```

```
    }

public static void main(String[] args){
Bbb b = new Bbb(18);
System.out.println(b.j +"\t" + b.k);
    }
}
```

**********回答问题**********

（1）执行 Bbb 的结果:

```
-1      18
```

（2）j = this.i;　　//本语句含义是：把 Baa 的变量 i 赋值给 j
k = super.i;　　//本语句含义是：把类 Aaa 的 i 给 k 赋值

（3）这个例子与上面例子的区别:

本例子中 Baa 中多了变量 i 可以隐藏从 Aaa 中继承过来的 i

3.对以下程序，运行程序回答问题。

```
classAaa{
void show(){
      System.out.println("Aaa 中定义的 show()");
  }
void show(inti){
      System.out.println("Aaa 中定义的 show(inti)");
  }
}

classBbb extends Aaa{
void show(){
      System.out.println("Bbb 中定义的 show()");
  }
public static void main(String[] args){
Aaa a = new Aaa();
Bbb b = new Bbb();
Aaa c = new Bbb();
a.show(1);
b.show(1);
c.show(1);
    }
}
```

**********回答问题**********

（1）执行 Bbb 的结果:

```
Aaa中定义的show(int i)
Aaa中定义的show(int i)
Aaa中定义的show(int i)
```

（2）上面 a.show(1)执行了那个类中定义的方法:

Aaa 定义的 show(int i)

（3）上面 b.show(1);执行了那个类中定义的方法:

Aaa 定义的 show(int i)

（4）上面 c.show(1)执行了那个类中定义的方法:

Aaa 定义的 show(int i)

（5）对程序执行结果的分析:

show(int i)只有在 Aaa 中有定义。

Aaa 中的 show()和 show(int i)重载，Bbb 中的 show()覆盖 Aaa 中的 show(),Bbb 继承了 Aaa 中的 show(i)

————————————————————————————————————

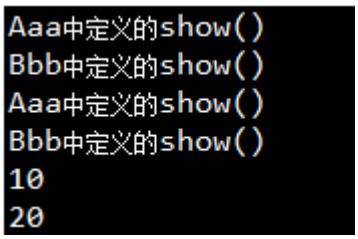
4. 对以下程序，回答问题。

```
classAaa{
inti = 10;
    static void show(){
     System.out.println("Aaa 中定义的 show()");
}
}

classBbb extends Aaa{
inti = 20;
static void show(){
      System.out.println("Bbb 中定义的 show()");
  }
public static void main(String[] args){
Aaa a = new Aaa();
Bbb b = new Bbb();
Aaa c = new Bbb();
a.show();
b.show();
c.show();
      ((Bbb)c).show();
System.out.println(c.i);     //考虑此处
System.out.println(((Bbb)c).i);     //考虑此处
   }
}
```

**********回答问题**********

执行 Bbb 的结果:



（1） ((Bbb)c).show()执行结果:

Bbb 中定义的 show()

（2）System.out.println(c.i)执行结果:

10

（3）System.out.println(((Bbb)c).i)执行结果:

20

（4）分析与结论:

通过引用变量 c 来访问变量或静态方法，要看 c 的声明类型,所以初始的 i 是 A 类的,在声明 Aaa c = new Bbb(); 直接输出 c.i 会显示类 Aaa 里初始化的结果。然后(Bbb)c).i 会强制输出类 Bbb 初始化的 i。

——

5.隐藏与覆盖的综合性实例

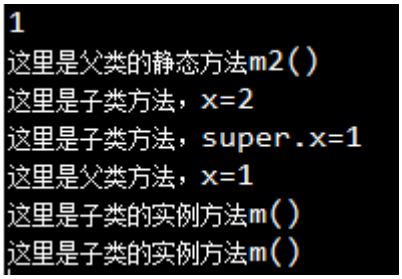
运行下面的程序，给出程序运行结果，理解掌握隐藏与覆盖

```
Class A{
 int x = 1;//被隐藏
 void print(){//被覆盖
        System.out.println("这里是父类方法，x="+x);//父类 A 的方法中访问的变量必然是 A 类或 A 的父类的，不可能访问 B 类的。
        m();//父类 A 的方法中调用的实例方法 m()是子类 B 的，由于发生了覆盖
 }
 void m(){//被覆盖
        System.out.println("这里是父类的实例方法 m()");
 }
 static void m2(){//被隐藏
        System.out.println("这里是父类的静态方法 m2()");
 }
}
class B extends A{
 int x = 2;
 void print(){
        System.out.println("这里是子类方法，x="+x);//子类方法访问的变量是子类对象的（当然条件是子类中声明了这个变量）
        System.out.println("这里是子类方法，super.x="+super.x);//super.x 是父类对象的
        super.print();//调用父类的 print()方法
        m();//调用本对象的 m()方法
 }
```

```
    void m(){
        System.out.println("这里是子类的实例方法 m()");
    }
    static void m2(){
        System.out.println("这里是子类的静态方法 m2()");
    }
}
public class TestEX01{
    public static void main(String []s){
        A p = new B();
        System.out.println(p.x);//通过引用变量 p 来访问变量或静态方法，要看 p 的声明类型。所以 x 是 A 类的。
        p.m2();//同上。静态方法 m2()是 A 类的。
        p.print();//通过引用变量 p 来访问实例方法，要看 p 指向的对象的实际类型。由于覆盖，调用的 print()方法是子类的。
    }
}
```

**********回答问题**********

1、给出程序执行结果



2、你对多态中“隐藏与覆盖”的理解

隐藏指的是子类把父类的属性或者方法隐藏了，即将子类强制转换成父类后，调用的还是父类的属性和方法，而覆盖则指的是父类引用指向了子类对象，调用的时候会调用子类的具体方法。

(1) 变量只能被隐藏(包括静态和非静态)，不能被覆盖

(3) 静态方法(static)只能被隐藏，不能被覆盖；

(3) 非静态方法可以被覆盖；

————————————————————————————————————

二、接口覆盖多态实验

请看下面的程序，然后回答问题。

```
interface IA{
public abstract void show();
}
class A implements IA{
public void show() {
    System.out.println("AAAA");
}
}
class B implements IA{
public void show() {
    System.out.println("BBBB");
```

```
}
}
public class Test{
public static void main(String[] args){
        IA[] a = new IA[4];
        a[0] = new A();
        a[1] = new B();
        a[2] = new B();
        a[3] = new A();
        for(inti=0;i<a.length;i++)
              a[i].show();
}
}
```

**********回答问题**********

（1）写出上述程序的运行结果。

```
AAAA
BBBB
BBBB
AAAA
```

（2）根据上述实验的结果，说明接口覆盖多态的特点。
继承接口的子类要重写覆盖的方法，这样才能调用自己的方法。

**********接口定义与实现**********

三、编写程序

学校中有老师和学生两类人，而在职研究生既是老师又是学生，对学生的管理和对教师的管理在他们身上都有体现。

1) 设计两个信息管理接口 StudentManageInterface 和 TeacherManageInterface。其中，StudentInterface 接口包括 setFee()方法和 getFee()方法，分别用于设置和获取学生的学费；TeacherInterface 接口包括 setPay()方法和 getPay()方法，分别用于设置和获取教师的工资

2) 定义一个研究生类 Graduate，实现 StudentInterface 接口和 TeacherInterface 接口，它定义的成员变量有 name(姓名)、xingbie(性别)、age(年龄)、fee(每学期学费)、pay(月工资)。

3) 创建一个姓名为“zhangsan”的研究生，统计他的年收入和学费，如果收入减去学费不足 2000 元，则输出“provide a loan”(需要贷款)信息。

提示：

1)定义两个接口，分别在其中申明两个方法。

2)定义主类 Graduate，实现这两个接口。

3)定义类 Graduate 的成员变量，和构造方法，并实现四个接口的方法，定义一个计算是否需要贷款的方法，在里面统计年收入和学费，并输出是否需要贷款的信息。

4)在 main 方法中创建一个姓名为“zhangsan”的研究 4 生，对 Graduate 类进行测试，并调用计算是否需要 6 贷款的方法。

**********回答问题**********

（1）给出上述程序代码

```
interface StudentManageInterface{
    public abstract void setFee(int fee);
    public abstract int getFee();
}

interface TeacherManageInterface{
    public abstract void setPay(int pay);
    public abstract int getPay();
}

class Graduate implements StudentManageInterface, TeacherManageInterface{
    int name, xingbie, age, fee, pay;
    public void setFee(int fee) { this.fee = fee; }
    public int getFee() { return fee; }
    public void setPay(int pay) { this.pay = pay; }
    public int getPay() { return pay; }
}

public class Demo {
    public static void main(String[] args){
        Graduate zhangsan = new Graduate();
        zhangsan.setFee(4000);
        zhangsan.setPay(5000);
        if (zhangsan.getPay() - zhangsan.getFee() < 2000)
            System.out.println("provide a loan");
    }
}
```

（2）给出上述程序运行结果

```
provide a loan
```

四、编写程序

1、定义接口：

```
Interface shape
{
float  PI =3.1415926;
    void  draw( );          //绘制图形，在程序中用信息输出表示
    double getArea( );      // 获取面积
   double  getCircumference ( )  // 周长
}
```

2、分别定义子类（继承接口 shape）：

圆（Circle）

椭圆（Eclipse）

矩形（Rectangle）

三角形（Triangle）

要求：

1）定义相应成员变量；

2）实现接口中定义的方法；

3）带参构造方法；

4）在程序中定义类 TestShape，并在 main 方法中完成对各类的测试

**********回答问题**********

（1）给出上述程序代码

```
interface shape{
   double  PI =3.1415926f;
    void  draw();          //绘制图形，在程序中用信息输出表示
    double getArea();      // 获取面积
    double  getCircumference();  // 周长
}

class Circle implements shape {
   double r;
   public Circle(double r) { this.r = r; }
   public void draw() {}
   public double getArea() { return shape.PI * r * r; }
   public double getCircumference() {
      return 2 * shape.PI * r * r;
   }
}

class Eclipse implements shape {
   double a, b;
   public Eclipse(double a, double b) { this.a = a; this.b = b; }
   public void draw() {}
   public double getArea() { return shape.PI * a * b; }
   public double getCircumference() {
      return 4 * ((a + b) - (4 - shape.PI) * a * b / (a + b));
```

```
    }
}

class Rectangle implements shape {
    double a, b;
    public Rectangle(double a, double b) { this.a = a; this.b = b; }
    public void draw() {}
    public double getArea() { return a * b; }
    public double getCircumference() {
        return 2 * (a + b);
    }
}

class Triangle implements shape {
    double a, b, c;
    public Triangle(double a, double b, double c) {
        this.a = a; this.b = b; this.c = c;
    }
    public void draw() {}
    public double getArea() {
        double p = (a + b + c)/2;
        return Math.sqrt(p*(p-a)*(p-b)*(p-c));
    }
    public double getCircumference() {
        return a + b + c;
    }
}

public class TestShape {
    public static void main(String[] args){
        Circle cir = new Circle(5.0);
        Eclipse ecl = new Eclipse(3.0, 4.0);
        Rectangle rec = new Rectangle(4.0, 5.0);
        Triangle tri = new Triangle(3.0, 4.0, 5.0);
        System.out.println("圆的面积: "+cir.getArea());
        System.out.println("圆的周长: "+cir.getCircumference());
        System.out.println("椭圆的面积: "+ecl.getArea());
        System.out.println("椭圆的周长: "+ecl.getCircumference());
        System.out.println("矩形的面积: "+rec.getArea());
        System.out.println("矩形的周长: "+rec.getCircumference());
        System.out.println("三角形的面积: "+tri.getArea());
        System.out.println("三角形的周长: "+tri.getCircumference());
    }
}
```

（2）给出上述程序运行结果

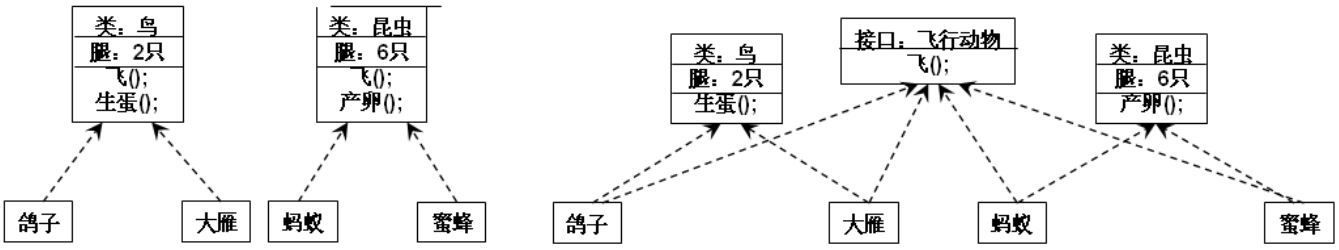
```
圆的面积：78.53981256484985
圆的周长：157.0796251296997
椭圆的面积：37.69911003112793
椭圆的周长：22.11377716064453
矩形的面积：20.0
矩形的周长：18.0
三角形的面积：6.0
三角形的周长：12.0
```

五、接口的用法

1、精简程序结构，免除重复定义

比如，有两个及上的的类拥有相同的方法，但是实现功能不一样，就可以定义一个接口，将这个方法提炼出来，在需要使用该方法的类中去实现，就免除了多个类定义系统方法的麻烦。举例：鸟类和昆虫类都具有飞行的功能，这个功能是相同的，但是其它功能是不同的，在程序实现的过程中，就可以定义一个接口，专门描述飞行。

分别定义鸟类和昆虫类，其都有飞行的方法；　　下图定义了接口，其类图如下：



要求： 根据上图，定义接口与类，并实例化相应类对象，实现输出。
代码：

```
interface Flyanimal {
    void fly();
}

class Insect {
    int leg = 6;
    void spawn() { System.out.println("产卵"); }
}

class Bird {
    int leg = 2;
    void layegg() { System.out.println("生蛋");}
}
```

```
class Pigeon extends Bird implements Flyanimal  {
   public void fly() { System.out.println("Pigeon can fly"); }
}

class Geese extends Bird implements Flyanimal {
   public void fly() { System.out.println("Geese can fly"); }
}

class Ant extends Insect implements Flyanimal {
   public void fly() { System.out.println("Ant can't fly"); }
}

class Bee extends Insect implements Flyanimal {
   public void fly() { System.out.println("Bee can fly"); }
}


public class Testanimal {
   public static void main(String[] args){
      Pigeon pig = new Pigeon();
      Geese  gee = new Geese();
      Ant an = new Ant();
      Bee be = new Bee();
      pig.fly(); pig.layegg();
      gee.fly(); gee.layegg();
      an.fly(); an.spawn();
      be.fly(); be.spawn();
   }
}
```

运行结果：

```
Pigeon can fly
生蛋
Geese can fly
生蛋
Ant can't fly
产卵
Bee can fly
产卵
```