实验三　继承性和多态

一、实验目的：熟悉 JAVA 中的继承性和多态性。掌握子类的定义及用法，继承机制中的隐藏与覆盖。子类的构造方法的用法，对象的上转型对象，抽象类与抽象方法等。

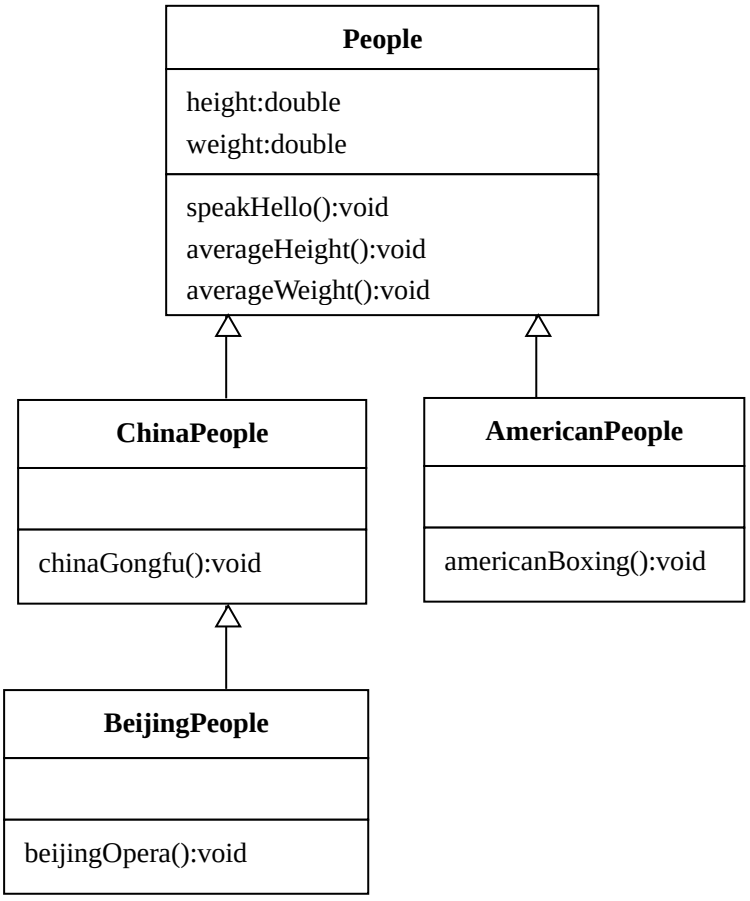
二、实验要求：

1．根据下面的要求，编写 Java 应用程序实现：

编写程序模拟中国人、美国人是人；北京人是中国人。除主类外，程序中还有 4 个类：People、ChinaPeople、AmericanPeople 和 BeijingPeople 类。要求如下：

- People 类有权限是 protected 的 double 型成员变量 height 和 weight，以及 public void speakHello()、public void averageHeight()和 public void averageWeight()方法。
- ChinaPeople 类是 People 的子类，新增了 public void chinaGongfu()方法。要求 ChinaPeople 重写父类的 public void speakHello()、public void averageHeight()和 public void averageWeight()方法。
- AmericanPeople 类是 People 的子类，新增 public void americanBoxing()方法。要求 AmericanPeople 重写父类的 public void speakHello()、public void averageHeight()和 public void averageWeight()方法。
- BeijingPeople 类是 ChinaPeople 的子类，新增 public void beijingOpera()方法。要求 ChinaPeople 重写父类的 public void speakHello()、public void averageHeight()和 public void averageWeight()方法。

People、ChinaPeople、AmericanPeople 和 BeijingPeople 类的 UML 图如下图所示：



在主类中定义各类的对象并调用其方法输出相应信息。

```java
package Main;
class People {
    protected double height;
    protected double weight;

    public void speakHello() {
        System.out.println("Hello");
    }

    public void averageHeight() {
    }

    public void averageWeight() {
    }
}
class ChinaPeople extends People {
    public void chinaGongfu() {

    }

    public void speakHello() {
        System.out.println("Hello");
```

```
    }

    public void averageHeight() {

    }

    public void averageWeight() {

    }
}
class AmercanPeople extends People {
    public void americanboxing() {

    }

    public void speakHello() {
        System.out.println("Hello");
    }

    public void averageHeight() {

    }
```

```
    public void averageWeight() {


    }
}
class BeijingPeople extends ChinaPeople {
    public void beijingOpera() {


    }
}
public class Main {
    static public void main(String argc[]) {
        People people = new People();
        ChinaPeople chinapeople = new ChinaPeople();
        AmercanPeople amercanpeople = new AmercanPeople();
        BeijingPeople beijingpeople = new BeijingPeople();
        people.speakHello();
        chinapeople.speakHello();
        amercanpeople.averageHeight();
        beijingpeople.chinaGongfu();
    }
}
```

2．根据下面的描述，编写 Java 程序实现：

假设银行 Bank 已经有了按整年 year 计算利息的一般方法，其中 year 只能取正整数。比如按整年计算的方法：

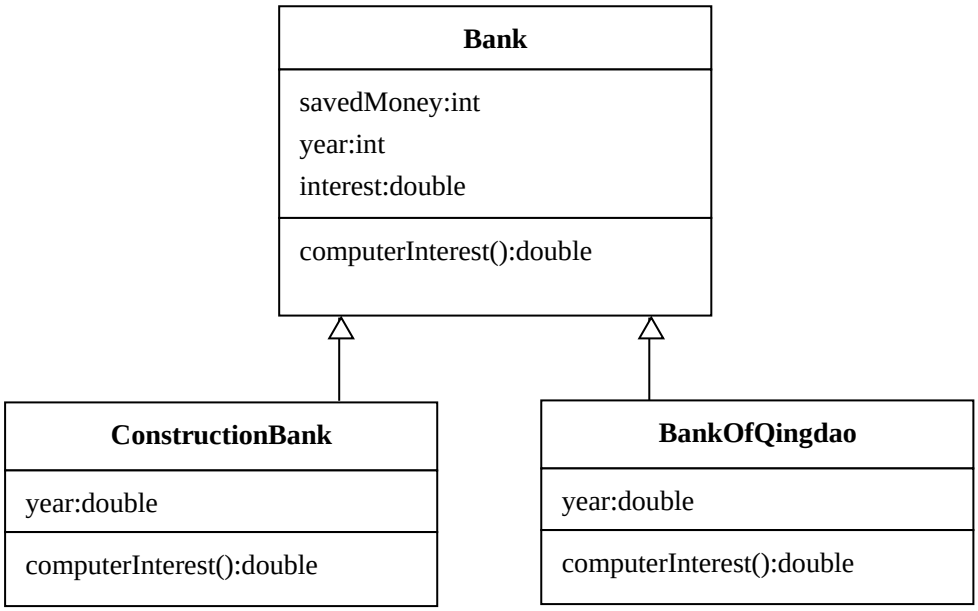
```
double computerInterest() {
    interest=year*0.035*savedMoney;
    return interest;
}
```

建设银行 ConstructionBank 是 Bank 的子类，准备隐藏继承的成员变量 year，并重写计算利息的方法，即自己声明一个 double 型的 year 变量，比如，当 year 取值为 5.216 时，表示要计算 5 年零 216 天的利息，但希望首先按银行 Bank 的方法 computerInterest()计算出 5 整年的利息，然后再自己计算 216 天的利息。那么，建设银行就必须把 5.216 的整数部分赋给隐藏的 year，并让 super 调用隐藏的、按整年计算利息的方法。

要求 ConstructionBank 和 BankOfQingdao 类是 Bank 类的子类，ConstructionBank 和 BankOfQingdao 都使用 super 调用隐藏的成员变量和方法。

计算 5 万元存 5 年零 216 天，在建设银行和青岛银行存的话，利息差额是多少？

ConstructionBank、BankOfQingdao 和 Bank 类的 UML 图如下所示：



注：整年利率：0.035 ，建设银行按天计算利率：0.0001，青岛银行按天计算利率：0.00015

功能扩展：

参照建设银行或青岛银行，再编写一个商业银行，让程序输出 8000 元存在商业银行 8 年零 236 天的利息。商业银行按天计算利率是 0.00012

```
class Bank {

    int year;
```

```
    int savedMoney;
    double interest;

    double computerInterest() {
        interest = year * 0.035 * savedMoney;
        return interest;
    }
}

class ConstructionBank extends Bank {
    double year;

    public ConstructionBank() {
        super.year = 5;
        super.computerInterest();
    }

    double computerInterest() {
        interest = super.interest + year * 0.0001 *
savedMoney;
        return interest;
    }
```

```
}

class BankOfQingdao extends Bank {
    double year;

    public BankOfQingdao() {
        super.year = 5;
        super.computerInterest();
    }

    double computerInterest() {
        interest = super.interest + year * 0.00015 *
savedMoney;
        return interest;
    }
}

class Businessbank extends Bank {
    double year;

    public Businessbank() {
        super.year = 8;
```

```
        super.computerInterest();
    }

    double computerInterest() {
        interest = super.interest + year * 0.00012 *
savedMoney;
        return interest;
    }
}

public class Main {
    static public void main(String argc[]) {
        Bank save = new Bank();
        save.year = 5;
        ConstructionBank constructionBank = new
ConstructionBank();
        BankOfQingdao bankOfQingdao = new BankOfQingdao();
        Businessbank businessbank = new Businessbank();
        businessbank.savedMoney = 8000;
        businessbank.year = 236;
        constructionBank.year = 216;
        bankOfQingdao.year = 216;
```

```
        constructionBank.savedMoney = 50000;
        bankOfQingdao.savedMoney = 50000;
        constructionBank.computerInterest();
        bankOfQingdao.computerInterest();
        businessbank.computerInterest();
        System.out.println(bankOfQingdao.interest -
constructionBank.interest);
        System.out.println(businessbank.interest);
    }
}
```

3. 根据下面要求，编写一个 Java 应用程序：

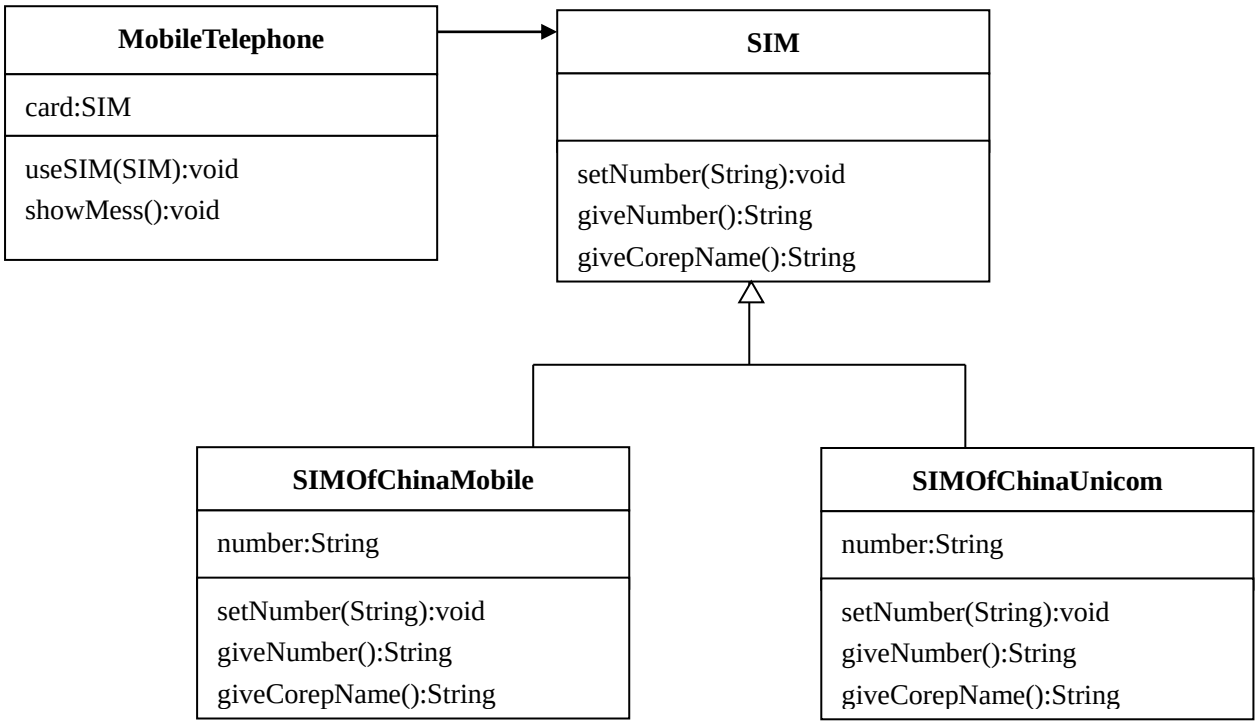
用类封装手机的基本属性和功能，要求手机即可以使用移动公司的 SIM 卡也可以使用联通公司 SIM 卡（可以使用任何公司提供的 SIM 卡）。

①.设计抽象类：设计一个抽象类 SIM，该类有三个抽象方法：giveNumber()、setNumber()和 giveCorpName()

②.设计手机类：设计 MobileTelephone，该类有 useSIM(SIM card)方法

③.各公司手机卡类：设计 SIMOfChinaMobile、SIMOfChinaUnicom 类

各类之间的关系如下：



编程定义各类，并在主类中进行测试。

```
abstract class SIM {
    public abstract String giveNumber();

    public abstract String giveCorpName();

    public abstract void setNumber(String n);
}
class SIMOFChinaMobile extends SIM {
    String phonenumber;

    public SIMOFChinaMobile() {
        phonenumber = "";
    }

    public SIMOFChinaMobile(String phonenumber) {
        this.phonenumber = phonenumber;
    }

    public String giveNumber() {
        return phonenumber;
```

```
    }

    public String giveCorpName() {
        return "中国移动";
    }

    public void setNumber(String phonenumber) {
        this.phonenumber = phonenumber;
    }
}
class SIMOFChinaUnicom extends SIM {
    String phonenumber;

    public SIMOFChinaUnicom() {
        phonenumber = "";
    }

    public SIMOFChinaUnicom(String phonenumber) {
        this.phonenumber = phonenumber;
    }

    public String giveNumber() {
```

```
        return phonenumber;
    }

    public String giveCorpName() {
        return "中国联通";
    }

    public void setNumber(String phonenumber) {
        this.phonenumber = phonenumber;
    }
}
class MobileTelephone {
    SIM card;

    void useSIM(SIM a) {
        card = a;
    }

    void showMess() {
        System.out.println("运营商:" +
card.giveCorpName());
        System.out.println("手机号:" + card.giveNumber());
```

```
    }
}
public class Main {
    static public void main(String[] argc) {
        SIM sim = new SIMOFChinaUnicom();
        sim.setNumber("18637780567");
        MobileTelephone phone = new MobileTelephone();
        phone.useSIM(sim);
        phone.showMess();
        sim = new SIMOFChinaMobile();
        sim.setNumber("17863976452");
        phone.useSIM(sim);
        phone.showMess();
    }
}
```

4. 根据下面要求，编写 Java 应用程序实现：

要求有一个 abstract 类，类名为 Employee。Employee 的子类有 YearWorker、MonthWorker、WeekWorker。YearWorker 对象按年领取薪水，MonthWorker 按月领取薪水，WeekWorker 按周领取薪水。Employee 类有一个 abstract 方法：

public abstract earnings();

子类必须重写父类的 earnings()方法，给出各自领取报酬的具体方式。

有一个 Company 类，该类用 Employee 对象数组作为成员，Employee 对象数组的元素可以是 YearWorker 对象的上转型对象、MonthWorker 对象的上转型对象或 WeekWorker 对象的上转型对象。程序能输出 Company 对象一年需要支付的薪水总额。

```
abstract class Employee {
    int money;
```

```
    public void earnings(int x) {
        money = x;
    }
}
class YearWorker extends Employee {
    int money;

    public void earnings(int x) {
        money = x;
    }
}
class MonthWorker extends Employee {
    int money;

    public void earnings(int x) {
        money = x;
    }
}
class WeekWorker extends Employee {
    int money;
```

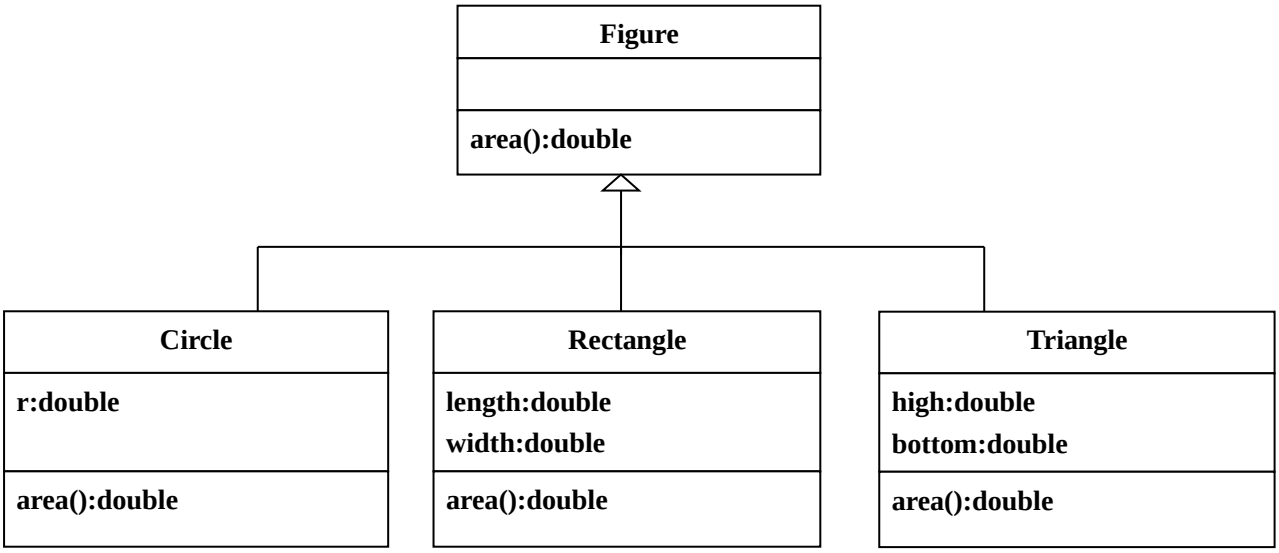
```
    public void earnings(int x) {
        money = x;
    }
}
public class Main {
    static public void main(String[] argc) {
        YearWorker yearworker = new YearWorker();
        MonthWorker monthworker = new MonthWorker();
        WeekWorker weekworker = new WeekWorker();
        yearworker.earnings(30);
        monthworker.earnings(20);
        weekworker.earnings(10);
    }
}
```

5. 对于各种几何图形，一般都有求图形的面积、周长等方法。现在有圆、矩形和三角形三种图形，要求通过类来实现求三种图形的面积。

问题分析：

三种图形都有相同的方法，因此可以抽象出一个抽象类：图形类，该类有抽象的求面积方法。然后由该抽象类生成 3 个子类：圆类、矩形类、三角形类。在主类中实现求各类图形的面积。

各类之间的关系如下：



在子类中，定义构造方法，实现对子类成员变量的初始化。
编程定义各类，并在主类中进行测试。

```
abstract class Figure {


    double area() {

        return 0;

    }

}

class Rectangle extends Figure {

    public double lenth, width;


    Rectangle(double l, double w) {

        lenth = l;

        width = w;
```

```
    }

    double area() {
        return lenth * width;
    }
}
class Triangle extends Figure {
    public double high, d;

    Triangle(double h, double dd) {
        high = h;
        d = dd;
    }

    double area() {
        return high * d / 2;
    }
}
class Circle extends Figure {
    public double r;

    Circle(double rr) {
```

```
        r = rr;
    }

    double area() {
        return r * r * 3.1415926;
    }
}
public class Main {
    static public void main(String[] argc) {
        Circle circle = new Circle(5);
        Triangle triangle = new Triangle(5, 5);
        Rectangle rectangle = new Rectangle(5, 5);
        System.out.println(circle.area());
        System.out.println(rectangle.area());
        System.out.println(triangle.area());
    }
}
```

6.设计一个动物声音“模拟器”，希望模拟器可以模拟许多动物的叫声，要求如下：

- 编写抽象类 Animal

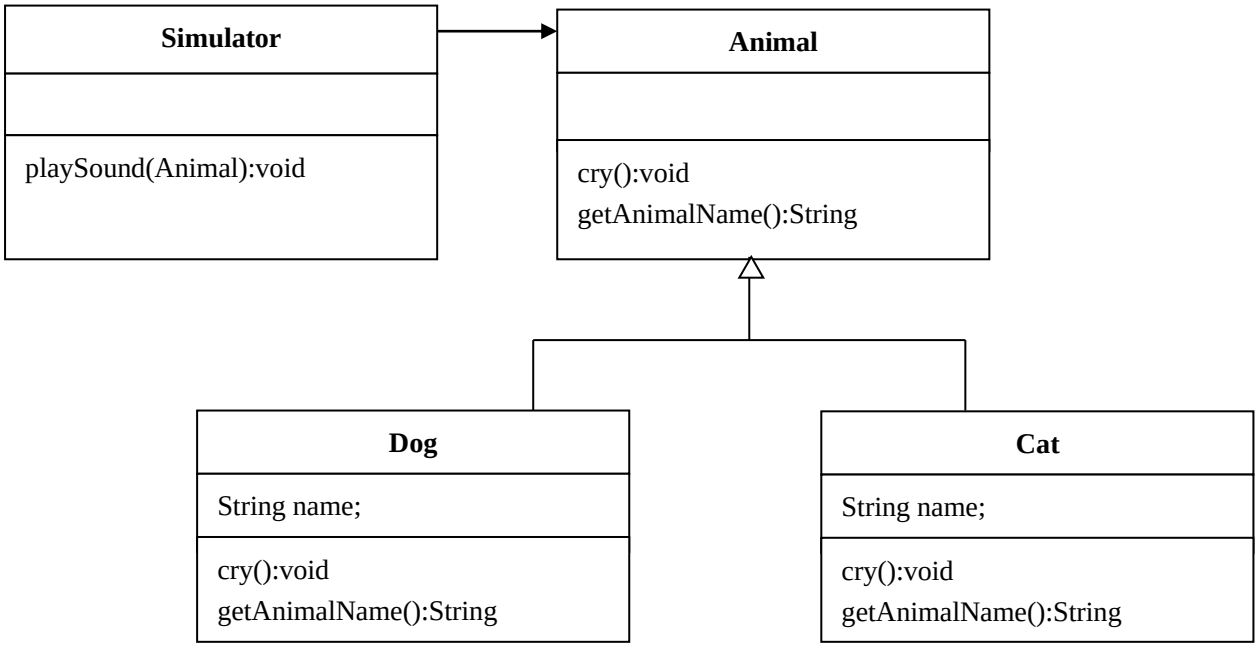
  Animal 抽象类有 2 个抽象方法 cry()和 getAnimalName()，即要求各种具体的动物给出自己的叫声和种类名称。

- 编写模拟器类 Simulator

  该类有一个 playSound(Animal animal)方法，该方法的参数是 Animal 类型。即参数 animal 可以调用 Animal 的子类重写的 cry()方法播放具体动物的声音，调用子类重写的 getAnimalName()方法显示动物种类的名称。

- 编写 Animal 类的子类：Dog 和 Cat 类

各类的 UML 图如下所示：



在各子类中通过构造方法实现对子类成员变量的初始化。

- 编写主类 Application（用户程序）

  在主类 Application 的 main 方法中至少包含如下代码。

  Simulator simulator = new Simulator();

  simulator.playSound(new Dog(“藏獒”));

  simulator.playSound(new Cat(“加菲猫”));

```
abstract class Animal {

    public String animalname;


    public void cry() {


    }


    public void getAnimalName(String name) {

        animalname = name;
```

```
    }
}
class Simulator {
    Animal playSound(Animal animal) {
        return animal;
    }
}
class Dog extends Animal {
    Dog(String name) {
        getAnimalName(name);
    }
}
class Cat extends Animal {
    Cat(String name) {
        getAnimalName(name);
    }
}
public class Application {
    static public void main(String[] argc) {
        Simulator simulator = new Simulator();
        simulator.playSound(new Dog("藏獒"));
        simulator.playSound(new Cat("加菲猫"));
```

```
    }

}
```