## 实验3.2 面向对象基础编程继承

|  |
| --- |
| 实验目的：   1. 理解继承的概念 2. 掌握继承的实现 3. 理解继承中的覆盖现象 |
| 实验数据记录及分析（或程序及运行结果）   1. 首先预测下列程序的运行结果，然后编译、执行以验证你的预测：   class Pet{  protected String name;  public Pet(String n){  name = n;  }  public String getName(){  return name;  }  public String move(){  return "run";  }  public String speak(){  return "";  }  public String toString(){  return "My pet " + name;  }  }  class Dog extends Pet{  protected int weight;  public Dog(String s){  super(s);  }  }  class PetTest{  public static void main(String[] args){  Pet myPet = new Pet("George");  Dog myDog = new Dog("Spot");  System.out.println(myPet.toString() + "\n" + "Speak: " +  myPet.speak() +"\n" + myPet.move() + " " +  myPet.getName() + "\n");  System.out.println(myDog.toString() + "\n" + "Speak: " +  myDog.speak() +"\n" + myDog.move() + " " +myDog.getName() + "\n");  }  }   * + 1. 修改Dog类的构造函数，使之可以初始化Dog类中的weight值。     2. 在Dog类中添加方法getWeight以获得其weight值。     3. 修改类Test，使用新的构造函数创建一个名为“snoopy”，重为10的Dog类的对象，并利用其方法输出包括重量在内的各项值     4. 要使Dog类对象的speak()方法返回值为“Woof Woof”,应如何做？修改（1）中程序，并验证你的做法。   1）**public** Dog(String s, **int** weight) {  **super**(s);  **this**.weight = weight;  }  2）**public** **int** getWeight() {  **return** weight;  }  3）Dog myDog = **new** Dog("snoopy", 10);  System.***out***.println(myDog.toString() + "\n" + "weight" + myDog.getWeight() + "\n" + "Speak: " + myDog.speak()  + "\n" + myDog.move() + " " + myDog.getName() + "\n");  4）**public** String speak() {  **return** "Woof Woof";  }   1. 按照要求完成程序   public class Test{  public static void main(String args[]){  System.out.println("创建父类对象：");  SuperClass sc0 = new SuperClass();  System.out.println("\n创建第一个子类对象：");  SubClass sc1 = new SubClass();  System.out.println("\n创建第二个子类对象：");  SubClass sc2 = new SubClass(1);  }  }  class SuperClass{  SuperClass( ){  System.out.println("父类的构造函数");  }  }  class SubClass extends SuperClass{  SubClass( ){  System.out.println("子类的第一个构造函数");  }  SubClass(int i){  System.out.println("子类的第二个构造函数");  }  **}**  要求：去掉SubClass的所有构造函数，然后将Test类中main方法的最后两行语句删除。预测修改后程序的运行结果，并进行验证。  **public** **class** Test {  **public** **static** **void** main(String args[]) {  System.***out***.println("创建父类对象：");  SuperClass sc0 = **new** SuperClass();  System.***out***.println("\n创建第一个子类对象：");  SubClass sc1 = **new** SubClass();  // System.out.println("\n创建第二个子类对象：");  // SubClass sc2 = new SubClass(1);  }  }  **class** SuperClass {  SuperClass() {  System.***out***.println("父类的构造函数");  }  }  **class** SubClass **extends** SuperClass {  // SubClass() {  // System.out.println("子类的第一个构造函数");  // }  //  // SubClass(int i) {  // System.out.println("子类的第二个构造函数");  // }  }  创建父类对象：  父类的构造函数  创建第一个子类对象：  父类的构造函数   1. **按照要求完成程序：** 2. 第一个类是图形类（Shape），含有一个成员变量color（字符串类型），一个没有参数的构造方法，以及一个有一个字符串类型参数的构造方法来初始化颜色变量，还有一个返回颜色变量值的成员方法show，以及一个方法getArea获取面积(不用具体代码实现)，返回值为double类型；   **public** **class** Shape {  **private** String color;  **public** Shape() {  **super**();  }  **public** Shape(String color) {  **super**();  **this**.color = color;  }  **public** **void** show() {  System.***out***.println("color:" + color);  }  **public** **double** getArea() {  **return** 1.1;  }  }   1. 第二个类是圆形类（Circle）继承自图形类，含有一个成员变量半径r，有一个有两个参数的构造方法，来初始化颜色和半径，一个方法getArea，返回值为double，获取圆的面积值。   **class** Circle **extends** Shape{  **private** **int** r;  **public** **static** **final** **double** ***PI***=3.14;    **public** Circle(String color,**int** r) {  **super**(color);  **this**.r = r;  }    **public** **double** getArea() {  **return** **this**.r\***this**.r\****PI***;  }  }   1. 第三个类是矩形类（Rectangle）继承自图形，含有两个double类型的成员变量长a和宽b，有一个有三个参数的构造方法，来初始化颜色、长和宽，一个方法getArea，返回值为double，获取矩形的面积值。   **class** Rectangle **extends** Shape {  **private** **double** a, b;  **public** Rectangle(String color, **double** a, **double** b) {  **super**(color);  **this**.a = a;  **this**.b = b;  }  **public** **double** getArea() {  **return** **this**.a \* **this**.b;  }  }   1. 第四个类是测试类（TestShape），分别定义圆形类和矩形类的实例对象，并用show方法，getArea方法来测试自己的定义   **class** TestShape {  **public** **static** **void** main(String[] args) {  Circle c = **new** Circle("红", 4);  Rectangle r = **new** Rectangle("蓝", 8, 5);  c.show();  System.***out***.println(c.getArea());  r.show();  System.***out***.println(r.getArea());  }  } |
| 实验小结：理解继承，学会使用super() |