# 实验四 接口与抽象类

【实验目的】

1. 掌握抽象类的应用
2. 掌握接口的定义与应用。

【实验要求】

1. 掌握创建接口的方法及接口的应用。

2. 掌握抽象类的创建和应用。

【实验内容】

1.下面有一个接口：

interface Shape

{

static double PI=3.14159

double area( );//计算面积

double volume( );//计算体积

}

请写出

（1）正方体、圆柱类、圆锥类（自己命名类名称），要求实现shape接口。

（2）为（1）中每个类编写构造方法。

（3）在（1）中每个类要求写出计算表面积和体积的方法。

（4）设计类 class Test ,要求在类Test中的主方法main中创建正方体，圆柱类，圆锥类的对象，并且计算表面积、体积并输出。

class Cone implements Shape {  
 private double radius;  
 private double height;  
  
 public Cone(double radius, double height) {  
 this.radius = radius;  
 this.height = height;  
 }  
  
 public double area() {  
 double slantHeight = Math.*sqrt*(radius \* radius + height \* height);  
 return *PI* \* radius \* (radius + slantHeight);  
 }  
  
 public double volume() {  
 return (*PI* \* radius \* radius \* height) / 3;  
 }  
}

class Cube implements Shape {  
 private double side;  
  
 public Cube(double side) {  
 this.side = side;  
 }  
  
 public double area() {  
 return 6 \* side \* side;  
 }  
  
 public double volume() {  
 return side \* side \* side;  
 }  
}

class Cylinder implements Shape {  
 private double radius;  
 private double height;  
  
 public Cylinder(double radius, double height) {  
 this.radius = radius;  
 this.height = height;  
 }  
  
 public double area() {  
 return 2 \* *PI* \* radius \* radius + 2 \* *PI* \* radius \* height;  
 }  
  
 public double volume() {  
 return *PI* \* radius \* radius \* height;  
 }  
}

public class Test {  
 public static void main(String[] args) {  
 Cube cube = new Cube(5.0);  
 System.*out*.println("Cube Area: " + cube.area());  
 System.*out*.println("Cube Volume: " + cube.volume());  
  
 Cylinder cylinder = new Cylinder(3.0, 4.0);  
 System.*out*.println("Cylinder Area: " + cylinder.area());  
 System.*out*.println("Cylinder Volume: " + cylinder.volume());  
  
 Cone cone = new Cone(2.0, 6.0);  
 System.*out*.println("Cone Area: " + cone.area());  
 System.*out*.println("Cone Volume: " + cone.volume());  
 }  
}

2. 抽象类：

（1）创建一个Vehicle类并将它声明为抽象类。在Vehicle类中声明一个NoOfWheels方法，使它返回一个字符串值。

（2）创建两个类Car和Motorbike从Vehicle类继承，并在这两个类中实现NoOfWheels方法。在Car类中，应当显示“四轮车”信息；而在Motorbike类中，应当显示“双轮车”信息。

（3）创建另一个带main方法的类，在该类中创建Car和Motorbike的实例，并在控制台中显示消息。

public abstract class Vehicle {  
 public abstract String NoOfWheels();  
}

public class Car extends Vehicle{  
 @Override  
 public String NoOfWheels() {  
 return "四轮车";  
 }  
}

public class Motorbike extends Vehicle{  
 @Override  
 public String NoOfWheels() {  
 return "双轮车";  
 }  
}

public class VehicleTest {  
 public static void main(String[] args) {  
 Car car = new Car();  
 Motorbike motorbike = new Motorbike();  
 System.*out*.println(car.NoOfWheels());  
 System.*out*.println(motorbike.NoOfWheels());  
 }  
}

3.练习教材例4-10~例4-20

public class Example10 {  
 static abstract class Animal{  
 abstract void shout();  
 }  
 static class Dog extends Animal{  
  
 @Override  
 void shout() {  
 System.*out*.println("汪汪");  
 }  
 }  
  
 public static void main(String[] args) {  
 Dog dog = new Dog();  
 dog.shout();  
 }  
}

public class Outer {  
 int m = 0;  
 void test1() {  
 System.*out*.println("外部成员方法 test1()");  
 }  
  
 class Inner {  
 int n = 1;  
 void show1() {  
 System.*out*.println("外部成员变量 m = " + m);  
 test1();  
 }  
  
 void show2() {  
 System.*out*.println("内部成员方法 show2()");  
 }  
  
 }  
  
 void test2() {  
 Inner inner = new Inner();  
 System.*out*.println("内部成员变量 n = " + inner.n);  
 inner.show2();  
 }  
}

public class Example20 {  
 public static void main(String[] args) {  
 Outer outer = new Outer();  
 Outer.Inner inner = outer.new Inner();  
 inner.show1();  
 outer.test2();  
 }  
}

【实验成绩】