# 实验7 抽象类与接口

**班级\_\_181\_\_\_\_\_\_\_\_\_学号1800502139姓名\_\_钟伟明\_**

**一、实验目的：**

1．掌握抽象类的概念、特点与定义方法。

2．掌握接口定义、特点与实现方法。

3．了解多接口的使用方法。

4．掌握接口与抽象类的区别。

**二、实验内容及要求：**

**1、掌握抽象类、抽象方法(完善程序)**

设计 3 个类 , 分别是学生类 Student, 本科生类 Undergaduate, 研究生类 Postgraduate, 其中 Student 类是一个抽象类 , 它包含一些基本的学生信息如姓名、所学课程、课程成绩等 , 而 Undergraduate 类和 Postgraduate 都是 Student 类的子类 , 它们之间的主要差别是计算课程成绩等级的方法有所不同 , 研究生的标准要比本科生的标准高一些 , 如表 1 所示。

**表 1 课程成绩等级**

|  |  |  |  |
| --- | --- | --- | --- |
| 本科生标准 | | 研究生标准 | |
| 80--100 | 优秀 | 90--100 | 优秀 |
| 70--80 | 良好 | 80--90 | 良好 |
| 60--70 | 一般 | 70--80 | 一般 |
| 50--60 | 及格 | 60--70 | 及格 |
| 50 以下 | 不及格 | 60 以下 | 不及格 |

假设某班级里既有本科生也有研究生 , 请编写程序统计出全班学生的成绩等级并显示出来。此题关键是设计一个学生数组 , 既能存放本科生对象 , 又能存放研究生对象。

(1) 仔细阅读程序清单后，补充空格中缺少的语句。

1) Student .java

**public** **abstract class** Student {

**final** **static** **int** *CourseNo* = 3;

**private** String name;

**private** String type;

**private** **int**[] courses;

**private** String courseGrade;

**public** Student(){ }

**public** Student(String name,String type)

{

**this**.name = name;

**this**.type = type;

courses = **new** **int**[*\_\_\_\_\_CourseNo \_\_\_\_\_*];//所学课程数组

courseGrade="";

}

**Public** abstract **void** calculateGrade();

**public** String getName( )

{

**return** name;

}

**public** String getType( )

{

**return** type; //返回学生类型

}

**public** String getCourseGrade( )

{

**return** courseGrade;

}

**public** **int** getCourseScore(**int** courseNumber)0

{

**return** courses[courseNumber];//c[0]

}

**public** **void** setName(String name)

{

**this**.name = name;

}

**public** **void** setType(String type)

{

**this**.type = type;

}

**public** **void** setCourseScore(**int** courseNumber, **int** courseScore)//0，1

{

Courses [courseNumber] = courseScore;//课程索引号设置课程成绩

}

**public** **void** setCourseGrade(String courseGrade) {

**this**.courseGrade = courseGrade;

}

}

2) Postgraduate .java

**class** Postgraduate **extends** Student {

**public** Postgraduate(String name)

{

**super**.Student(name,"研究生");

}

**public** **void** calculateGrade()

{

**int** total = 0;

**double** average = 0;

**for** (**int** i = 0; i < *CourseNo*; i++) {

total =total + getCourseScore(i);

};

average = total / *CourseNo*;

String currentGrade="";

**if** (average>=90&&average<=100){

currentGrade = "优秀";

}

**else** **if** (average>=80&&average<90) {

currentGrade = "良好";

}

**else** **if** (average>=70&&average<80) {

currentGrade = "一般";

}

**else** **if** (average>=60&&average<70) {

currentGrade = "及格";

}

**else**{

currentGrade = "不及格";

}

setCourseGrade(currentGrade);

}

}

3 Undergraduate.java

**class** Undergraduate **extends** Student {

**public** Undergraduate(String name)

{

**super**(name,”本科生”);

}

**public** **void** calculateGrade() {

**int** total = 0;

**double** average = 0;

**for** (**int** i = 0; i < *CourseNo*; i++) {

total=total+getCourseScore(i); // 累加各门课程成绩

};

average = total / *CourseNo*;

String currentGrade="";

**if** (average>=80&&average<100){

currentGrade = "优秀";

}

**else** **if** (average>=70&&average<80){

currentGrade = "良好";

}

**else** **if** (average>=60&&average<70){

currentGrade = "一般";

}

**else** **if** (average>=50&&average<60){

currentGrade = "及格";

}

**else**{

currentGrade = "不及格";

}

setCourseGrade(currentGrade);

}

}

4) PolymorphismTest.java

**import** java.util.arry;

**public** **class** PolymorphismTest {

**public** **static** **void** main(String[] args) {

Student studentArray = **new** Student[5];

studentArray [0] = **new** Undergraduate("张三");//本科生张三

studentArray [1] = **new** Undergraduate("李四"); //本科生李四

studentArray [2] = **new** Postgraduate("王五"); //研究生王五

studentArray [3] = **new** Undergraduate("赵六"); //本科生赵六

studentArray [4] = **new** Postgraduate("钱七"); //研究生钱七

**for** (**int** i=0; i<studentArray.length;i++) {

studentArray[i].setCourseScore(0,87);

studentArray[i].setCourseScore(1,90);

studentArray[i].setCourseScore(2,78);

}

**for** (**int** i=0; i<5 ;i++) {

studentList[i].calculateGrade();

}

System.*out*.println("姓名" + " 类型" +" 成绩");

System.*out*.println("-----------------------");

**for** (**int** i=0; i<5 ;i++) {

System.*out*.println(studentArray[i].getName( )+" "+

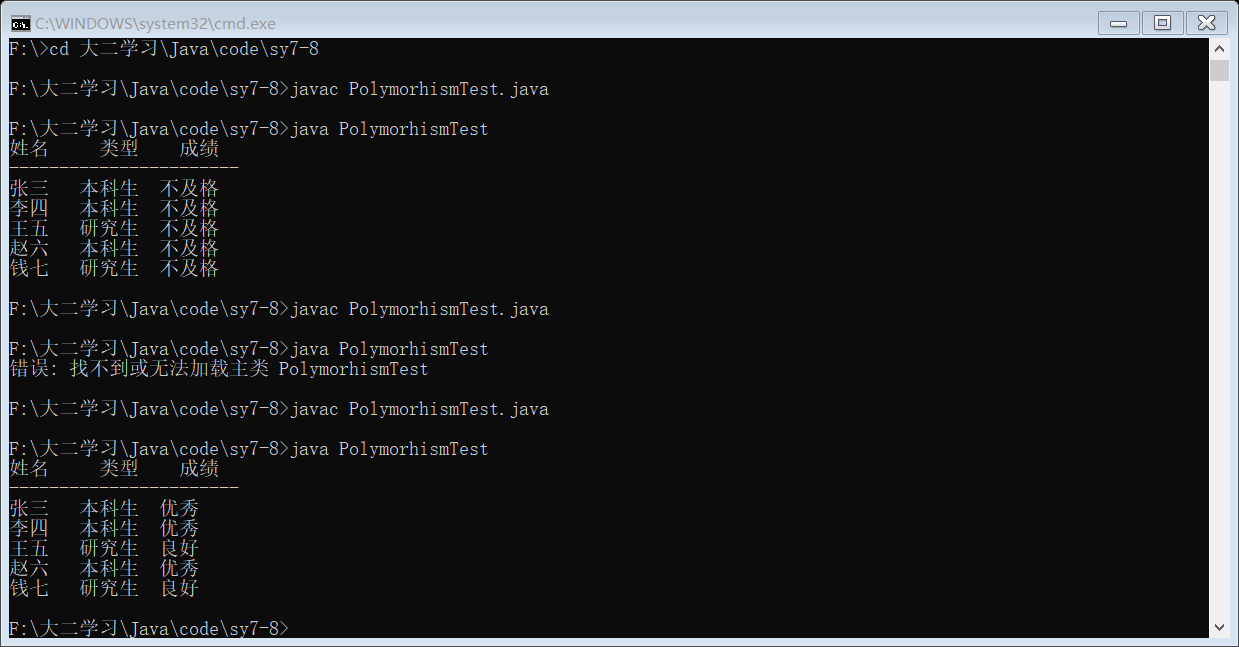
studentArray[i].getType( )+" "+

studentArray[i].getCourseGrade( ));

}

}

}



**程序2. 定义一个抽象类（Shape)实现如下功能**

abstract class Shapes {

public int x,y; //x，y为的坐标

public int width,height;

public Shapes(int x,int y,int width,int height) {

............................

}

public abstract double getArea();

public abstract double getPerimeter();

}

Class Rectangle extends Shape{

...................

}

Class Circle extends Shape{

Double radius;//半径

...................

}

//矩形类（Rectangle）需要实现返回面积的方法（getArea）和计算周长的方法(Perimeter)，//圆形（Circle）自行扩充成员变量和方法。

import java.lang.Math;

abstract class Shapes {

public int x,y; //x，y为的坐标

public int width,height;

public Shapes(int x,int y,int width,int height) {

this.x= x;

this.y= y;

this.width= width;

this.height= height;

}

public abstract double getArea();

public abstract double getPerimeter();

}

class Rectangle extends Shapes{

public Rectangle(int x,int y,int width,int height) {

super(x,y,width, height);

}

public double getArea() {

return width\*height;

}

public double getPerimeter() {

return 2\*(width+height);

}

}

class Circle extends Shapes{

double radius,radius2;//半径

public Circle(int x,int y,int radius,int radius2) {

super(x,y,radius, radius2);

}

public double getArea() {

return Math.PI\*super.height\*super.width;

}

public double getPerimeter() {

return Math.PI\*super.height\*super.width;

}

}

class AreaPerimeter{

public static void main(String[] args) {

Circle cir = new Circle(0,0,5,5);

System.out.println(cir.getArea());

}

}

**程序3：**

**（1）源程序**

import java.util.Scanner;

interface Soundable {

public void increaseVolume( );

public void decreaseVolume( );

public void stopSound( );

public void playSound( );

}

class Radio implements Soundable {

public void increaseVolume( ) {

System.out.println("增大收音机音量");

}

public void decreaseVolume( ) {

System.out.println("减小收音机音量");

}

public void stopSound( ) {

System.out.println("关闭收音机");

}

public void playSound( ) {

System.out.println("收音机播放广播");

}

}

class Walkman implements Soundable {

public void increaseVolume( ) {

System.out.println("增大随声听音量");

}

public void decreaseVolume( ) {

//代码1 输出减小随声听音量

}

public void stopSound( ) {

System.out.println("关闭随声听");

}

public void playSound( ) {

System.out.println("随声听发出音乐");

}

}

class Mobilephone implements Soundable {

public void increaseVolume( ) {

System.out.println("增大手机音量");

}

public void decreaseVolume( ) {

System.out.println("减小手机音量");

}

public void stopSound( ) {

System.out.println("关闭手机");

}

public void playSound( ) {

System.out.println("手机发出来电铃声");

}

}

class People {

private String name;

private int age;

public void listen(Soundable s) {

s.playSound( );

}

}

public class InterfaceTest {

public static void main(String[] args) {

int i;

People sportsman = new People( );

Scanner scanner = new Scanner(System.in);

Soundable[] soundDevice = new Soundable[3];

//往声音设备数组中放入能发声的设备

soundDevice[0] = new Radio( );

soundDevice[1] = new Walkman( );

代码2 //创建手机对象并赋值给soundDevice[2]

System.out.println("你想听什么? 请输入选择：0-收音机 1-随声听 2-手机");

i = scanner.nextInt( );

//开始听声音

sportsman.listen(soundDevice[i]);

soundDevice[i].increaseVolume( );

代码3 //调用stopSound( )方法

}

}

**(2) 六个java源文件，放在同一个文件夹中。**

**(3) 编译之后运行这个程序 , 观察所得结果。**

**思考：**

1. 请问在 InterfaceTest 类中，SoundDevice[] 数组是什么类型的 ，该数组为什么能存放 3 种不同的对象 Radio、Walkman 和 Mobilephone 呢？

是Soundable 接口类型

因为Radio、Walkman 和 Mobilephone实现了接口

(2) 在程序中，Soundable 是一个接口，那么该接口是否可以被实例化呢？请在InterfaceTest 类的 main() 方法中加入以下语句试验一下 , 并分析结果。

Soundable Sound=new Soundable(),

接口不可以被实例化

(3) 现在假定要为程序增加一个闹钟类 Clock，该类也实现 Soundable 接口，能够发出滴答声，请将以下的 Clock 类加入到 InterfaceTest.java 程序中，并在 InterfaceTest 类的 main() 方法中加入 SoundDevice[3] =new Clock(); 语句。

class Clock implements Soundable{

public void stopSound(){

System.out.println(" 关闭闹钟 ");

}

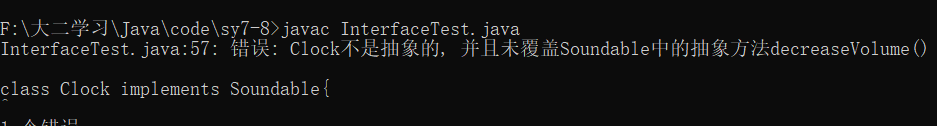
public void playSound(){

System.out.println(" 闹钟发出滴答声 ");

}

}

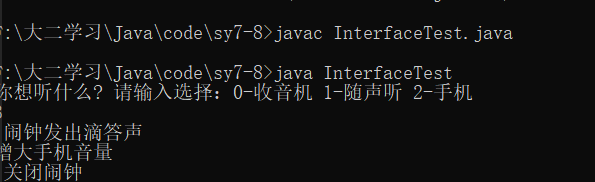
修改之后 , 重新编译 InterfaceTest.java 并运行它 , 观察结果。



(4) 由于新加入的 Clock 类仅仅实现了 Soundable 接口的stopsound() 和 playsound() 方法，而 increaseVolume() 和 decreaseVolume() 方法没有实现，因此它实质上是一个抽象类，而抽象类是不能实例化的，所以导致编译错误。但是按照常理，闹钟的滴答声确实是不可以增大或减小的，那么如何解决这个问题呢？现在请在 Clock 类中加入下面两个含 {} 空方法体的方法实现，再编译运行程序，看看会有什么变化。

public void increaseVolume(){}

public void decreaseVolume(){}



(5) 现在请模仿本实验的程序设计出一个自己的接口程序 , 要求先设计一个 moveable 可移动接口(声明向前、后、左、右方法), 然后分别设计 3 个类 , 即汽车 Car 、轮船 Ship 、飞机 Aircraft 来实现该接口 , 最后设计一个应用程序来使用它们。

import java.util.Scanner;

interface Moveable{

public void ahead();

public void back();

public void turnleft();

public void turnright();

}

class Car implements Moveable{

public void ahead(){

System.out.println("汽车向前");

}

public void back() {

System.out.println("汽车后退");

}

public void turnleft() {

System.out.println("汽车向左");

}

public void turnright() {

System.out.println("汽车向右");

}

}

class Ship implements Moveable{

public void ahead(){

System.out.println("轮船向前");

}

public void back() {

System.out.println("轮船后退");

}

public void turnleft() {

System.out.println("轮船向左");

}

public void turnright() {

System.out.println("轮船向右");

}

}

class Aircraft implements Moveable{

public void ahead(){

System.out.println("飞机向前");

}

public void back() {

System.out.println("飞机后退");

}

public void turnleft() {

System.out.println("飞机向左");

}

public void turnright() {

System.out.println("飞机向右");

}

}

class Control {

public void move(Moveable m) {

m.ahead();

}

}

public class MoveableTest {

public static void main(String[] args) {

int i,j;

Control pilot = new Control();

Scanner scanner = new Scanner(System.in);

Moveable[] Device = new Moveable[3];

Device [0] = new Car();

Device [1] = new Ship();

Device [2] = new Aircraft();

System.out.println("你想操纵什么? 请输入选择：0-汽车 1-轮船 2-飞机 ");

i=scanner.nextInt();

pilot.move(Device[i]);

System.out.println("你想怎么操纵? 请输入选择：0-前进 1-后退 2-向左 3-向右 ");

j=scanner.nextInt();

switch(j) {

case 0:Device[i].ahead();break;

case 1:Device[i].back();break;

case 2:Device[i].turnleft();break;

case 3:Device[i].turnright();break;

default:break;

}

}

}

**点评：**接口的特点在于只定义能做什么，而不定义怎么去做。在本实验中，收音机 Radio，随身听 Walkman 和手机 Mobilephone 分别以自己的方式实现了 Soundable 接口，当接口成为 Listen(Soundable s) 方法的形参时，任何实现了 Soundable 接口的对象都能成为它的实参，如果不用接口作形参，那就必须写 3 个不同的方法， 即

listenRadio(Radio r),

istenWalkman(Walkman w) ，

listenMobilephone(Mobilephone m)。

**（三）存在的问题、解决办法及小结**

本次实验对于抽象类有了更深的理解，继承了抽象类的方法必须把抽象类的方法全部实现