电子科技大学信息与软件工程学院

**实 验 报 告**

学 号 2017221103015

姓 名 张磊

（实验） 课程名称 类的继承、接口及访问控制

理论教师 周帆

实验教师 何中海

**电子科技大学教务处制表**

**电 子 科 技 大 学**

**实 验 报 告**

**学生姓名：张磊 学号：2017221103015 指导教师：何中海**

**实验地点：信软楼西304 实验时间：2018.12.23**

**一、实验名称：类的继承、接口及访问控制**

**二、实验学时：2**

**三、实验目的：熟悉Java的类的继承基本操作，接口及访问控制，理解面向对象程序设计的基本思想**

**四、实验原理：**

继承：类继承另一个类，这个类除了创建自己的成员外，还能够继承或扩展另一个类的成员。运用继承，你能够创建一个通用类，它定义了一系列相关项目的一般特性。该类可以被更具体的类继承，每个具体的类都增加一些自己特有的东西。被继承的类叫超类（superclass），继承超类的类叫子类（subclass）。

继承性特性如下：

1) 单一继承性：子类只能有一个超类，而超类可以有多个子类；

2) 子类继承超类的所有成员；

3) 子类可以创建自己的成员；

4) 子类不能继承超类的构造器，只能在构造器中通过super()调用超类的构造器；

5) 子类的构造器首先要调用超类的构造器；

6) 多态性之一：子类的成员隐藏和覆盖超类中相同的成员；

7) 多态性之二：超类的对象可以对子类的实例引用；

8) 由abstract和final修饰的类指示该类是否必须或不能被继承。

创建子类，格式如下：

[abstract|final] class SubCls extends SupCls{

SubClassBody

}

abstract是可选项，修饰的类叫抽象类，指示其对象引用的必须是其子类实例；

final是可选项，修饰的类叫终结类，指示其不能被继承，不能有子类；

SubCls是创建的类，称为子类；

extends是关键字，指示两个类存在的继承关系；

SuperCls是SubCls类的超类。

Super:

在子类中，使用super访问被隐藏的超类变量，被覆盖的超类方法。

使用有三种情况：

1) 访问被隐藏的超类成员变量；

例如： super.varName

2) 调用超类中被覆盖的方法；

例如： super.methodName([paramList])

3) 调用超类中的构造方法.

例如：super([paramList])

This:

通过this调用另一个构造方法，用法是this(参数列表)，这个仅仅在类的构造方法中，别的地方不能这么用。在函数中，需要引用该函数所属类的当前对象时候，直接用this（访问当前对象）。

函数参数或者函数中的局部变量和成员变量同名的情况下，成员变量被屏蔽，此时要访问成员变量则需要用“this.成员变量名”的方式来引用成员变量。当然，在没有同名的情况下，可以直接用成员变量的名字，而不用this。--2的特例

运行时的多态:

超类的对象可以引用子类的实例（向上转型），该对象仍然只能够调用超类中已定义的方法和变量对于覆盖或继承的方法，Java运行时系统根据调用该方法的实例的类型来决定选择哪个方法调用。

对子类的一个实例：

1) 如果子类覆盖了超类的方法，则运行时系统调用子类的方法。

2) 如果子类继承了超类的方法(未覆盖)，则运行时系统调用超类的方法。

访问控制：

public：可访问性最大修饰符，由public修饰的成员，则可以被任何范围中所访问。

protected：允许类中、子类(包括在或不在同一包中)和它所在包中的类所访问。

缺省：可以被类自身和同一个包中的类访问。

private：限制最强的修饰符。私有成员只能在它自身的类中访问。可以最大限度地保持好类中敏感变量和方法，避免对象对这些类的成员访问时带来危害。

接口：

Java是通过接口使得处于不同类层次，甚至互不相关的类可以具有相同的行为。接口是方法定义（没有实现， java8 之后可以有static方法和default 方法）、常量的集合。用接口，你可以指定一个类必须做什么，而不是规定它如何去做。在类层次的任何地方都可以使用接口定义一个行为的协议实现它。

接口的定义格式与类相似，具有成员变量和成员方法。但是接口中的所有方法都是abstract方法，这些方法是没有语句的。 （java8 之后可以有static方法和default 方法）

格式：

interface Declaration{

interfaceBody

}

其中：

interfaceDeclaration为接口声明部分；

interfaceBody为接口体部分。

接口的实现：

1) 接口的方法必须由(非抽象的)类非抽象实现。

2) 可以由抽象类抽象的实现

3) 在类的声明中，如果用implements子句就可以声明这个类对接口的实现。

4) 关键字implements不同于extends，它表示类对接口的实现而不是继承，并且一个类可以实现多个接口。

5) 类实现接口，则必须实现接口中的所有方法。

6) 实现接口时，不能降低接口中的可访问性（必须是public，因为接口中方法都是public）

接口的类型：

接口可以作为一个引用类型来使用。任何实现该接口的类的实例都可以存储在该接口类型的变量中，通过这些变量可以访问这个类所实现接口中的方法。在程序运行时，Java动态地确定需要使用哪个类中的方法。

类的继承中，超类变量引用子类对象——上转型(upcasting)

接口实现中，接口变量引用子类对象——回调(callback)

接口可以通过运用关键字extends被其他接口继承。语法与继承类是一样的。当一个类实现一个继承了另一个接口的接口时，它必须实现接口继承链表中定义的所有方法。

接口中的变量和方法的被隐藏和覆盖：如果在子接口中定义了和超接口同名的常量或相同的方法，则超接口中的常量被隐藏，方法被覆盖。

**五、实验内容：**

1. 完成第五章习题9、10编程。

2. 完成第六章习题7、8编程。

3. 有几何形状边数为n及可计算面积area的Shape类，其子类Triangle类及Rectangle类实现几何形状三角形和矩形面积area计算，利用前三个形状类实现柱体Pillar类的体积计算，并在PillarTest类中实现对某一柱体的体积计算。

4. 创建学生成绩中所涉及的类：Student类、Teacher类、Course类，并由Grade类将Student类、Teacher类和Course类关联起来，由GradeTest类对以上四个类进行测试。

**六、实验器材（设备、元器件）：**

装有eclipse软件的PC机一台

**七、实验步骤：**

1. 明确项目需求
2. 编写代码
3. 编译代码
4. 测试程序

5) 根据测试结果对程序进行调试改进

**八、实验结果与分析（含重要数据结果分析或核心代码流程分析）**

第五章习题9代码如下：

import java.util.Scanner;

public class Test5\_9{

public static void main(String []args){

Person person1=new Person("张磊",'男',19);

System.out.println("person1:"+person1.getDate());

Person person2=person1;

person2.setDate("李四",'男',34);

System.out.println("person2:"+person2.getDate());

Student student1=new Student("王五",'女',17);

System.out.println("student1:"+student1.getDate());

Student student2=student1;

student2.setDate("翠花",'女',17,001,1103);

System.out.println("student2:"+student2.getDate());

}

}

class Person{

String name;

char sex;

int age;

Person(String n,char s,int a){

name=n;

sex=s;

age=a;

}

void setDate(String n,char s,int a){

name=n;

sex=s;

age=a;

}

String getDate(){

return "name:"+name+" sex:"+sex+" age:"+age;

}

}

class Student extends Person{

int sID,classNo;

Student(String n,char s,int a){

super(n,s,a);

}

void setDate(String n,char s,int a,int i,int c){

super.setDate(n,s,a);

sID=i;

classNo=c;

}

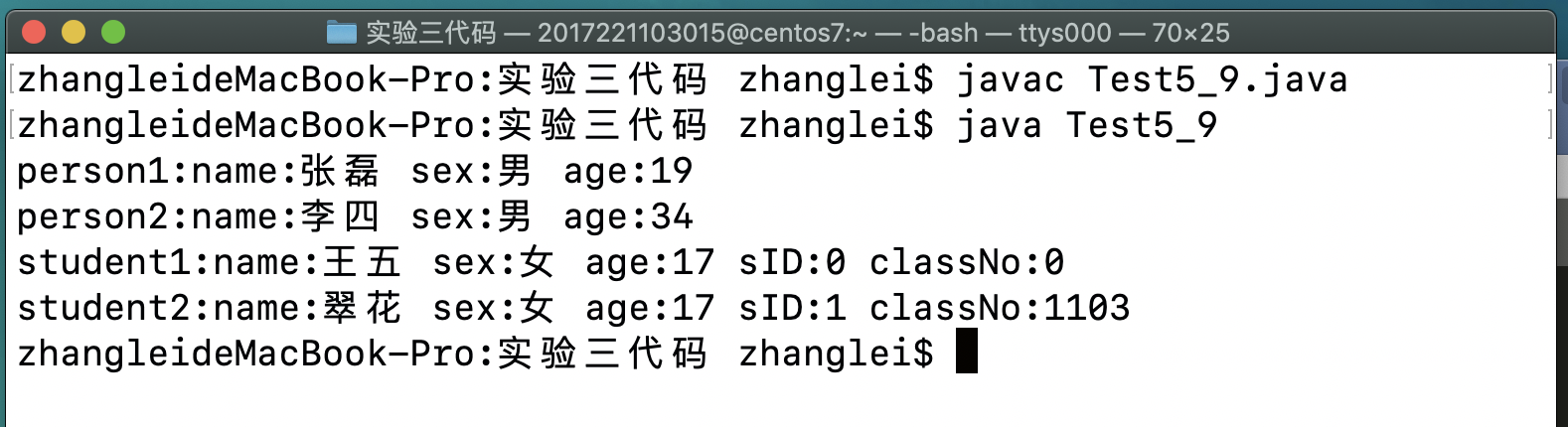
String getDate(){

return super.getDate()+" sID:"+sID+" classNo:"+classNo;

}

}

第五章习题9运行结果如图1-1

图1-1

第五章习题10代码如下：

public class Test5\_10{

public static void main(String []args){

Teacher teacher=new Teacher();

teacher.setDate("王老师",'女',29);

System.out.println("teacher:"+teacher.getDetail());

Student student=new Student();

student.setDate("张三",'男',14);

System.out.println("student:"+student.getDetail());

}

}

abstract class Person{

String name;

char sex;

int age;

abstract void setDate(String name,char sex,int age);

abstract String getDetail();

}

class Teacher extends Person{

String name;

char sex;

int age;

int tID;

String department;

void setDate(String n,char s,int a){

name=n;

sex=s;

age=a;

}

String getDetail(){

return "name:"+name+" sex:"+sex+" age:"+age+" tID:"+tID+" department:"+department;

}

}

class Student extends Person{

String name;

char sex;

int age;

int sID;

String speciality;

void setDate(String n,char s,int a){

name=n;

sex=s;

age=a;

}

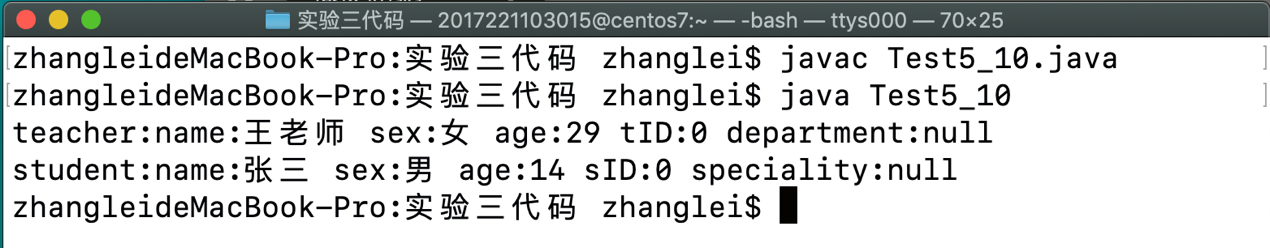
String getDetail(){

return "name:"+name+" sex:"+sex+" age:"+age+" sID:"+sID+" speciality:"+speciality;

}

}

第五章习题10程序运行结果如图1-2所示

图1-2

第六章习题7代码如下所示：

public class Test6\_7{

public static void main(String []args){

classA person=new classA();

person.setDate("Sam",'m',19);

person.print();

classB student=new classB();

student.setDate("Jim","math",95.5);

student.print();

}

}

interface Print{

void print();

}

class classA implements Print{

String name;

char sex;

int age;

void setDate(String n,char s,int a){

name=n;

sex=s;

age=a;

}

public void print(){

System.out.println("name:"+name+" sex:"+sex+" age:"+age);

}

}

class classB implements Print{

String name;

String subject;

double grade;

void setDate(String n,String sub,double g){

name=n;

subject=sub;

grade=g;

}

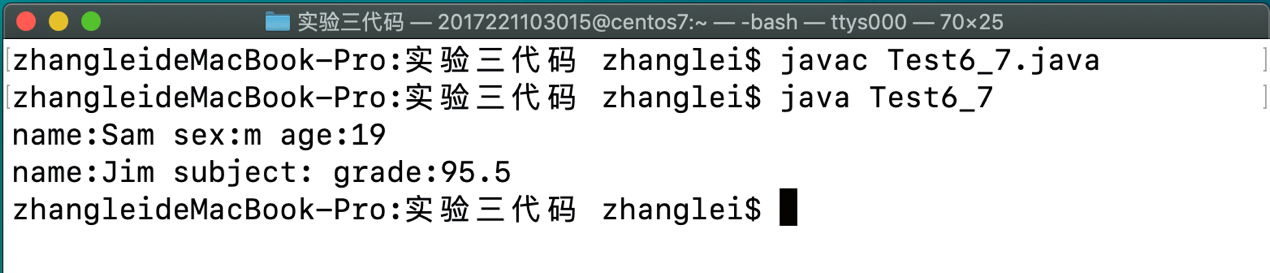
public void print(){

System.out.println("name:"+name+" subject:"+" grade:"+grade);

}

}

第六章习题7程序运行结果如图1-3所示

图1-3

第六章习题8代码如下：

public class Test6\_8{

public static void main(String []args){

Student student=new Student();

student.setDate(015,"software");

System.out.println(student.getDate());

}

}

interface Person{

String name="Jimmy";

char sex='w';

int birthday=19990303;

void setDate(int s,String str);

String getDate();

}

class Student implements Person{

int sID;

String speciality;

public void setDate(int s,String spec){

sID=s;

speciality=spec;

}

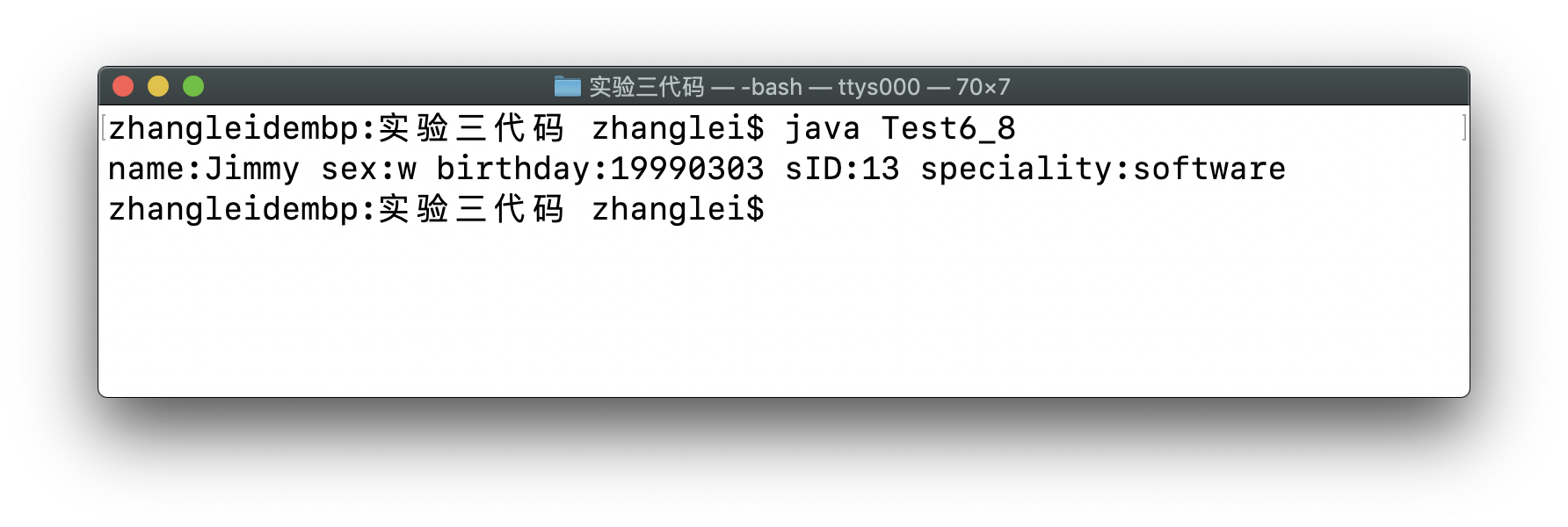
public String getDate(){

return "name:"+name+" sex:"+sex+" birthday:"+birthday+" sID:"+sID+" speciality:"+speciality;

}

}

第六章习题6程序运行结果如图1-4所示

图1-4

实验内容3代码如下所示：

import java.util.Scanner;

public class PillarTest{

public static void main(String []args){

String str;

double h;

System.out.println("请输入柱体的高：");

h=new Scanner(System.in).nextDouble();

System.out.println("请输入柱体底面的形状（三角形/矩形）：");

str=new Scanner(System.in).nextLine();

if (str.equals("三角形")) {

Pillar pillar=new Pillar(3,h);

pillar.volume=pillar.getVolume();

System.out.println("该三棱柱的体积为："+pillar.volume);

}else if (str.equals("矩形")) {

Pillar pillar=new Pillar(4,h);

pillar.volume=pillar.getVolume();

System.out.println("该四棱柱的体积为："+pillar.volume);

}else{

System.out.println("抱歉，输入错误，您只能选择三角形或者矩形。");

System.exit(1);

}

}

}

abstract class Shape{

double area;

abstract double getArea();

}

class Triangle extends Shape{

double width,length;

Triangle(double w ,double l){

width=w;

length=l;

}

public double getArea(){

return width\*length;

}

}

class Rectangle extends Shape{

double length1,length2,length3;

Rectangle(double l1,double l2,double l3){

length1=l1;

length2=l2;

length3=l3;

}

public double getArea(){

double p=0.5\*(length1+length2+length3);

double area\_2=p\*(p-length1)\*(p-length2)\*(p-length3);

return java.lang.Math.sqrt(area\_2);

}

}

class Pillar{

int n;

double highth;

Shape shape;

double volume;

Pillar(int n,double highth){

this.n=n;

this.highth=highth;

}

double getVolume(){

if(n==3){

System.out.println("请输入底面三角形的三边长：");

double l1=new Scanner(System.in).nextDouble();

double l2=new Scanner(System.in).nextDouble();

double l3=new Scanner(System.in).nextDouble();

shape=new Rectangle(l1,l2,l3);

shape.area=shape.getArea();

return shape.area\*highth;

}else{

System.out.println("请输入底面矩形的长和宽：");

double l1=new Scanner(System.in).nextDouble();

double l2=new Scanner(System.in).nextDouble();

shape=new Triangle(l1,l2);

shape.area=shape.getArea();

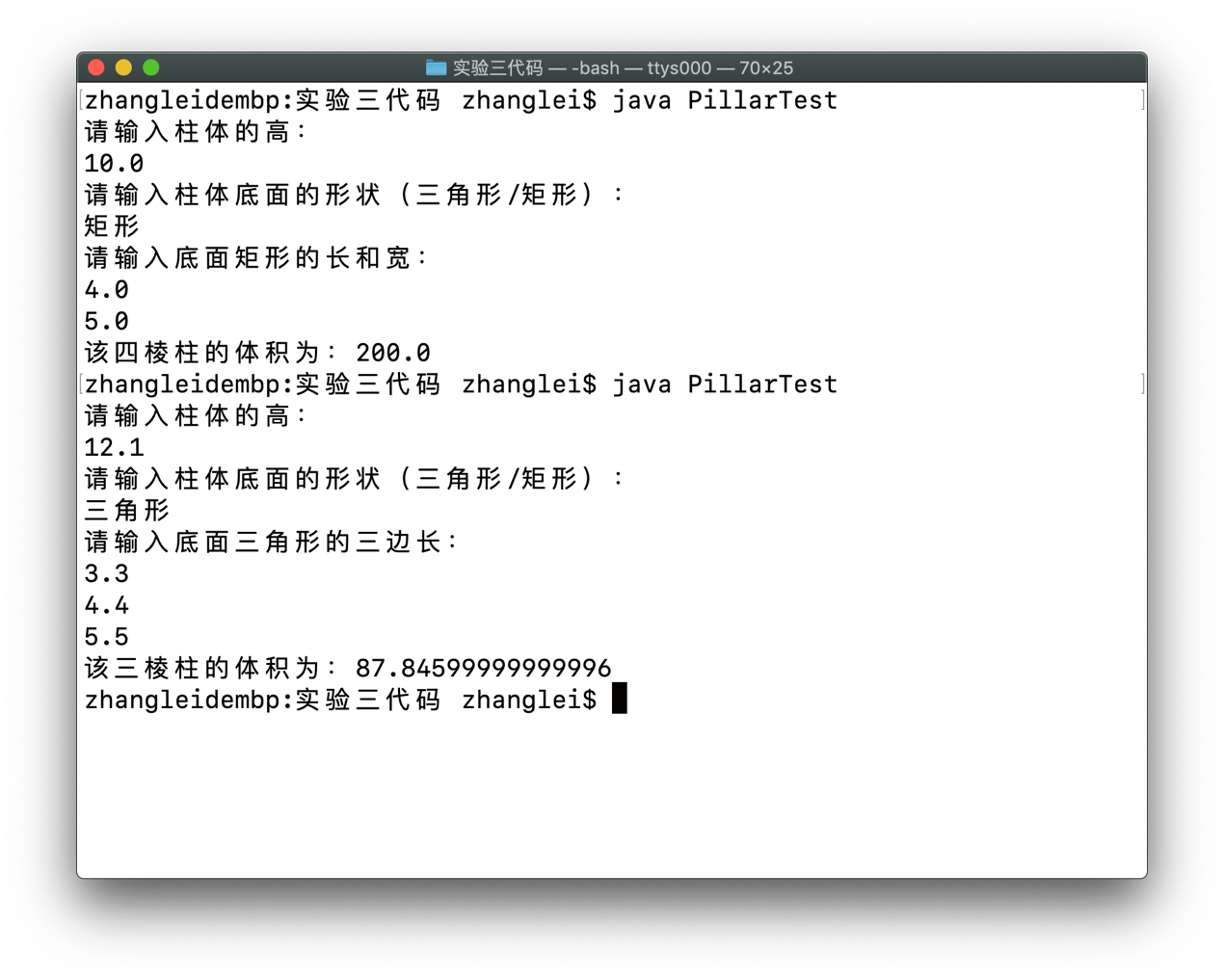
return shape.area\*highth;

}

}

}

实验内容3程序运行结果如图1-5所示

图1-5

实验内容4代码如下：

import java.util.Scanner;

public class GradeTest{

public static void main(String []args){

System.out.println("开始录入成绩信息：");

Grade grade=new Grade();

grade.begin();

System.out.println("录入完毕\n录入的成绩信息如下：\n课程名称："+grade.course.course+" 学生："+grade.student.name+" 任课教师："+grade.teacher.name+" 成绩："+grade.grade);

}

}

class Grade{

double grade;

Student student=new Student();

Teacher teacher=new Teacher();

Course course=new Course();

void begin(){

System.out.println("请输入课程成绩：");

grade=new Scanner(System.in).nextDouble();

student.setDate(teacher,course);

teacher.setDate(student,course);

course.setDate(student,teacher);

}

}

class Student{

Teacher teacher;

String name;

Course course;

void setDate(Teacher t,Course course){

teacher=t;

this.course=course;

System.out.println("请输入学生姓名：");

name=new Scanner(System.in).nextLine();

}

}

class Teacher{

Student student;

String name;

Course course;

void setDate(Student stu,Course cour){

student=stu;

course=cour;

System.out.println("请输入教师姓名：");

name=new Scanner(System.in).nextLine();

}

}

class Course{

String course;

Teacher teacher;

Student student;

void setDate(Student stu,Teacher tea){

student=stu;

teacher=tea;

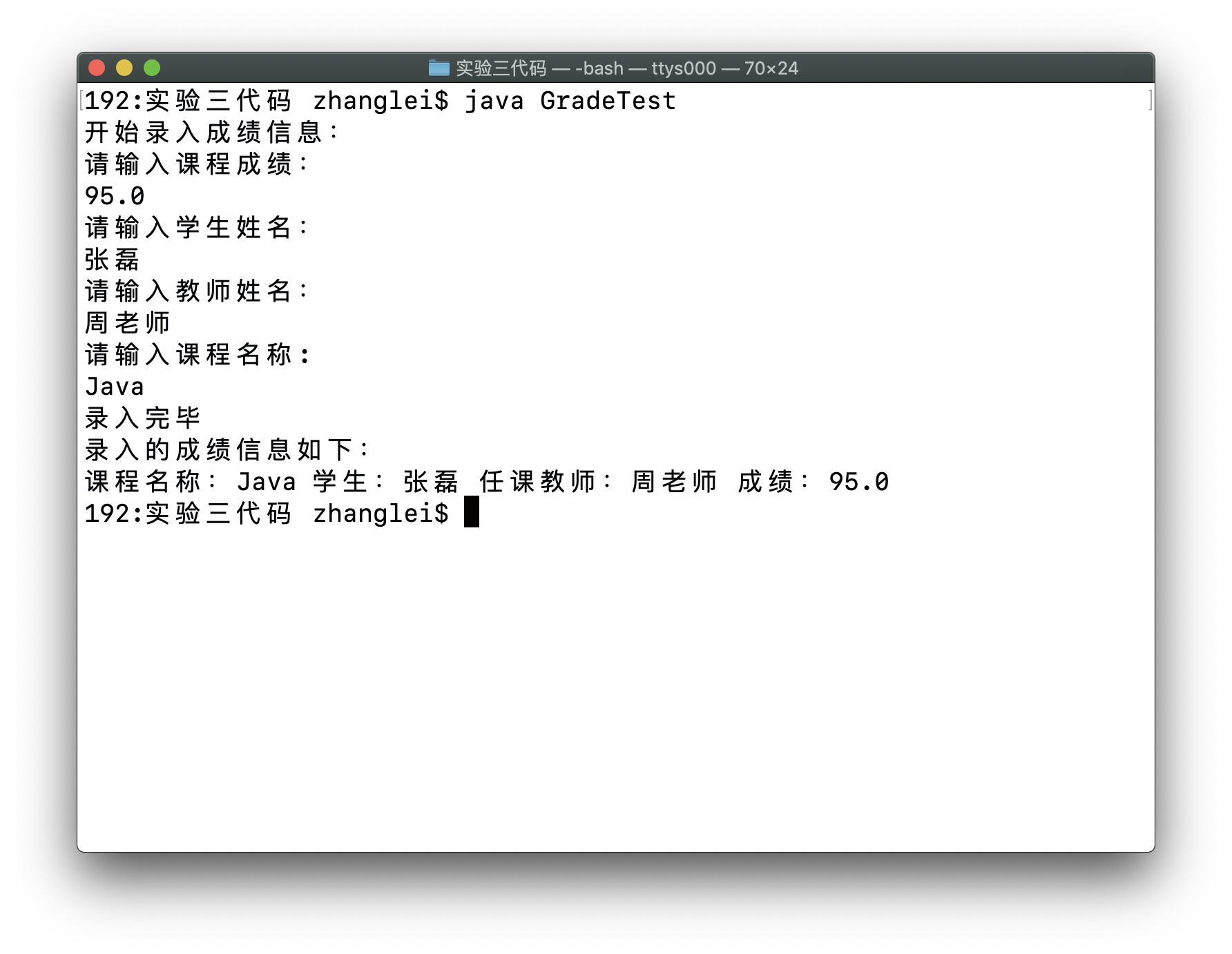
System.out.println("请输入课程名称:");

course=new Scanner(System.in).nextLine();

}

}

实验内容4程序运行结果如图1-6所示

图1-6

**九、总结及心得体会：**

在编写代码的时候，尽量将具体的实现方法都封装到函数中去，只暴露出需要的借口，这样可以降低代码之间的耦合度，使得代码的可复用性更强，代码质量更高。在获取学生信息时，最开始我直接在getData()方法的return语句中进行字符串的连接，但其实创建一个tostring方法还更增加代码可读性，而且eclipse有创建tostring方法的快捷键，使用起来相对方便些，在实现接口的类中，复写接口中的抽象方法时，开始总是忘记public修饰符，因为接口里的方法默认时public、static、abstract。而类中的方法访问权限不能比它的低，所以要带上public，不能为缺省值。在理解内容三和四的问题上，刚开始对题意不是很明确，不知道具体需要完成什么功能，运行什么方法，让我花费了一些时间，可能是对“关联”二字理解不够深刻。写这次实验的代码时，又让我记起访问方法不仅可以用实例化的对象名去访问，还可以用新创建的实例直接访问方法。

做完这次实验，感觉对继承的总体掌握还比较浅，只会运用其一些基本的特点，还需细细去品味！

**十、对本实验过程及方法、手段的改进建议：**

在写代码时，尽量使不同的两个类相似的地方少，即减少代码量，能用一个父类或接口替代的，就不要让代码看着感觉比较啰嗦

**报告评分：**

**指导教师签字：**