**《面向对象程序设计》实验报告**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **课程名称** | | | **《面向对象程序设计》** | | | | |
| **实验名称** | | | **实验三 类和对象** | | | | |
| **姓名** | | 卓如意 | | **学号** | 22145138 | **班级** | 计算机科学与技术221 |
| **实**  **验**  **目**  **的** | 1. 初步理解面向对象的基本概念：封装、类、对象，方法，成员变量和局部变量，构造方法等； 2. 能根据应用需求发现、设计、实现和使用类、对象，管理对象集合； 3. 培养良好的编程规范。 | | | | | | |
| **实**  **验**  **内**  **容** | 1. 结合PTA在线练习6学习静态成员/实例成员、构造函数等基本概念； 2. 结合PTA在线练习7掌握对象间的交互操作 3. 模拟实现一个小型的学生选课系统，每个学生可以选一些课程，并可获得查询每门课的成绩等信息，按照平均绩点对学生进行比较等。实现学生类、课程类、控制类的定义。具体要求请查看附件中的各类的描述及参考的测试代码。 | | | | | | |
| **实验过程记录** | 1. 静态成员/实例成员、构造函数等基本概念  （1）静态成员：  静态成员变量：也称为类变量。这种类型的变量属于整个类而不是单个对象。因此，所有对象共享相同的静态变量值。静态变量存储在程序的数据区域，生命周期始于类加载时，结束于类卸载时。  静态方法：也称为类方法。这类方法可以直接通过类名调用，而不必创建类的实例。静态方法不能访问非静态成员变量，因为静态方法是在类加载时就存在的，而非静态成员变量只有在类实例化时才会被分配内存。  （2）实例成员：  实例成员变量：也称为对象变量。每个类的实例都有自己的副本，这意味着不同对象的实例成员变量可以有不同的值。  实例方法：定义在类中的非静态方法。实例方法可以访问实例成员变量和静态成员变量，但不能直接访问其他实例的方法，除非通过引用传递。  （3）构造函数：  构造函数是一种特殊的方法，用于初始化新创建的对象。它的名称必须与类名相同，并且没有返回类型。每当创建一个对象时，都会自动调用相应的构造函数来设置对象的状态。  public class MyClass {  int myVariable;  // 默认构造函数  public MyClass() {  this.myVariable = 0;  }  // 带参数的构造函数  public MyClass(int value) {  this.myVariable = value;  }  }  在这个例子中，MyClass有两个构造函数：一个默认构造函数和一个带参数的构造函数。当创建一个MyClass对象时，编译器会根据传入的参数选择合适的构造函数进行初始化。  构造函数调用规则：  如果你没有显式地编写任何构造函数，Java将为你提供一个默认的无参构造函数。  如果你提供了至少一个构造函数，Java不会为你提供默认的构造函数。  子类不会继承父类的构造函数，但是可以通过super关键字调用父类的构造函数。  构造函数不能被子类重写，也不能被直接调用。  构造函数的第一个语句通常是对超类构造函数的调用，这被称为构造函数链。  2. 在线练习7  （1）7-1 构造方法  源代码：  public class Main {  public Main(){  System.out.println("构造方法一被调用了");  }  public Main(int x){  this();  System.out.println("构造方法二被调用了");  }  public Main(boolean b){  this(1);  System.out.println("构造方法三被调用了");  }  public static void main(String[] args) {  Main s1=new Main(true);  }  }  （2）7-2 声明图书类，记录图书总册数，利用静态变量赋值。  源代码：  public class Main{  public static void main(String[] ss){  Tushu[] s1=new Tushu [3];  s1[0]=new Tushu("Java程序设计","1",34.5);  s1[1]=new Tushu("数据结构","2",44.8);  s1[2]=new Tushu("C++程序设计","3",35.0);  int sum=0;  for(int i=0;i<3;i++){  System.out.println(s1[i]);  sum+=s1[i].getCount();  }  System.out.println("图书总册数为："+sum);  }  }  class Tushu{  private String name;  private String bh;  private double price;  private int count;  public Tushu(String name,String bh,double price){  setName(name);  setBh(bh);  setPrice(price);  setCount(1);  }  public void setName(String name){  this.name=name;  }  public void setBh(String bh){  this.bh=bh;  }  public void setPrice(double price){  this.price=price;  }  public void setCount(int count){  this.count=count;  }  public String getName(){  return this.name;  }  public String getBh(){  return this.bh;  }  public double getPrice(){  return this.price;  }  public int getCount() {  return this.count;  }  public String toString(){  return "书名："+name+", 书号："+bh+", 书价："+price;  }  }  （3）7-3 jmu-Java-03面向对象基础-02-构造方法与初始化块  源代码：  import java.util.Scanner;  public class Main{  public static void main(String [] ss){  Scanner sc=new Scanner(System.in);  int n= sc.nextInt();  Person [] a=new Person[n];  for(int i=0;i<n;i++){  String name=sc.next();  int age=sc.nextInt();  boolean gender=sc.nextBoolean();  a[i]=new Person(name,age,gender);  }  for(int i=n-1;i>=0;i--){  System.out.println(a[i]);  }  Person b=new Person();  System.out.println(b.getName()+","+b.getAge()+","+b.getGender()+","+b.getId());  System.out.println(b);  }  }  class Person{  private String name;  private boolean gender;  private int age;  private static int count=0;  private int id=count++;  public Person(){  System.out.println("This is constructor");  }  public Person(String name,int age,boolean gender) {  this.setName(name);  this.setGender(gender);  this.setAge(age);  }  {  System.out.println("This is initialization block, id is " + id);  }  static {  System.out.println("This is static initialization block");  }  public void setName(String name){  this.name=name;  }  public void setGender(boolean gender){  this.gender=gender;  }  public void setAge(int age){  this.age=age;  }  public void setId(int id){  this.id=id;  }  public String getName(){  return name;  }  public boolean getGender(){  return gender;  }  public int getAge(){  return age;  }  public int getId(){  return id;  }  @Override  public String toString() {  return "Person [name=" + name + ", age=" + age+", gender=" + gender +  ", id=" + id + "]";  }  }  （4）7-4 统计商品总价  源代码：  import java.util.Scanner;  class Main{  public static void main(String args[]){  Goods ga[] =new Goods[5];  Scanner sc = new Scanner(System.in);  for(int i =0;i<5;i++){  ga[i]= new Goods(sc.next(),sc.nextDouble());  }  double shouldPay = 0;  for(Goods g:ga){  shouldPay += g.getPrice();  System.out.println(g.toString());  }  System.out.println("should pay:"+shouldPay);  }  }  class Goods{  private String name;  private double price;  public Goods(String name,double price){  this.setName(name);  this.setPrice(price);  }  public void setName(String name){  this.name=name;  }  public void setPrice(double price){  this.price=price;  }  public String getName(){  return name;  }  public double getPrice(){  return price;  }  @Override  public String toString(){  return name+","+price;  }  }  （5）7-5 设计一个风扇Fan类  源代码：import java.util.\*;  class Fan {  public final int SlOW=1;  public final int MEDIUM=2;  public final int FAST=3;  private int speed=SlOW;  private boolean on=false;  private double radius=5;  private String color="white";  Fan(){}  Fan(int fanSpeed,boolean fanOn,double fanRadius,String fanColor){  speed=fanSpeed;  on=fanOn;  radius=fanRadius;  color=fanColor;  }  public String toString() {  String s="speed "+speed+"\n";  s+="color "+color+"\n";  s+="radius "+radius+"\n";  if(on==false) {  s+="fan is off";  }else {  s+="fan is on";  }  return s;  }  }  public class Main {  public static void main(String[] args) {  System.out.println("-------\n"  + "Default\n"  + "-------");  Fan fan1=new Fan();  System.out.println(fan1.toString());  System.out.println("-------\n"  + "My Fan\n"  + "-------");  Scanner in=new Scanner(System.in);  int fanSpeed=in.nextInt() ;  boolean fanOn=in.nextBoolean();  double fanRadius=in.nextDouble();  String fanColor=in.next();  Fan fan2=new Fan(fanSpeed, fanOn,fanRadius,fanColor);  System.out.println(fan2.toString());  }  }  （6）7-6 sdust-Java-学生成绩读取与排序  源代码：  import java.util.\*;  class Student implements Comparable<Student>{  private String name,id;  private int subject=1;  private int score;  private double sum=0;  Student(){}  Student(String name,String id,int score){  this.name=name;  this.id=id;  this.score=score;  }  public String getid(){  return this.id;  }  public void subjectadd() {  this.subject++;  }  public void scoreadd(int score){  this.score=this.score+score;  }  public String getname() {  return this.name;  }  public void sum() {  this.sum=this.score/this.subject;  }  public int compareTo(Student o1){  Student one = (Student) o1;  if(this.sum-one.sum!=0)  return (int)(one.sum-this.sum);  else  return this.id.compareTo(one.id);  }  }  public class Main{  public static void main(String[] args) {  Scanner sc=new Scanner(System.in);  List<Student> list =new ArrayList<Student>();  int i,flag=0;  String k;  String[] and =new String[5];  while(sc.hasNext())  {  k=sc.next();  and=k.split(",");  if(k.compareTo("exit")==0)  break;  for(i=0;i<list.size();i++)  {  if(list.get(i).getid().compareTo(and[1])==0)  {  flag=1;  break;  }  }  if(flag==1)  {  list.get(i).subjectadd();  list.get(i).scoreadd(Integer.parseInt(and[3]));  }  else  {  list.add(new Student(and[0],and[1],Integer.parseInt(and[3])));  }  flag=0;  }  for(i=0;i<list.size();i++)  list.get(i).sum();  Collections.sort(list);  for(i=0;i<list.size();i++){  System.out.println("No"+(i+1)+":"+list.get(i).getid()+","+list.get(i).getname());  }  }  }  3. 小型的学生选课系统  （1）源代码：  import javax.management.MXBean; import java.util.ArrayList; import java.util.Arrays;  class Course {  String name;  int grade;  double points;   public Course(String name, int grade, double points) {  this.name = name;  this.grade = grade;  this.points = points;  }   public int getGP() {  if (grade >= 90) return 4;  if (grade >= 80) return 3;  if (grade >= 70) return 2;  if (grade >= 60) return 1;  return 0;  }  public boolean hasGrade(){  return grade!=0;  } }  class Student {  String name;  ArrayList<Course> myCourse;   public Student(String name) {  this.name = name;  this.myCourse = new ArrayList<>();  }   public Student(String name, Course[] courses) {  this.name = name;  this.myCourse = new ArrayList<>(Arrays.*asList*(courses));  }   public int courseGradeSum() {  int sum = 0;  for (Course c : myCourse) {  sum += c.grade;  }  return sum;  }   public double courseEveGrade() {  return (double) courseGradeSum() / myCourse.size();  }   public double getGPA() {  double sumCredit = 0;  double sumGradePoint = 0;  for (Course c : myCourse) {  sumCredit += c.points;  sumGradePoint += c.getGP() \* c.points;  }  return sumGradePoint / sumCredit;  }   public boolean isBetterThan(Student s) {  return this.getGPA() > s.getGPA();  }   public static Student whoIsBest(Student s1, Student s2) {  return s1.isBetterThan(s2) ? s1 : s2;  }   public static Student whoIsBest(Student[] ss) {  Student best = ss[0];  for (int i = 1; i < ss.length; i++) {  if (!best.isBetterThan(ss[i])) {  best = ss[i];  }  }  return best;  }   public void addCourse(Course c) {  if (!myCourse.contains(c)) {  myCourse.add(c);  } else {  System.*out*.println("该课程已存在，无法再次选择");  }  }   public void dropCourse(Course c) {  if (myCourse.contains(c) && !c.hasGrade()) {  myCourse.remove(c);  } else {  System.*out*.println("该课程不存在或已有成绩，无法退选");  }  } }  class Test {  public static void main(String[] args) {  Course[] cs1 = {new Course("数据结构", 80, 5), new Course("数据库", 90, 3),new Course("面向对象", 80, 3)};  Course[] cs2 = {new Course("软件工程", 60, 5), new Course("数据库", 80, 3),new Course("面向对象", 80, 3)};  Course[] cs3 = {new Course("数据结构", 90, 5), new Course("数据库", 80, 3),new Course("软件工程", 90, 3),new Course("面向对象", 90, 3)};  Student []s={new Student("John", cs1),new Student("Mike", cs2),new Student("Rose", cs3)};  System.*out*.println(Student.*whoIsBest*(s).name + "的总成绩为：" + Student.*whoIsBest*(s).courseGradeSum());  System.*out*.println(Student.*whoIsBest*(s).name + "的平均绩点为：" + Student.*whoIsBest*(s).getGPA());   } }  （2）不足：  只在内部测试，没有实现外部实现数据的增删查改 | | | | | | |
| **实验心得**  **体会** | （通过本次实验，谈谈你对面向对象编程的理解，包括封装的意义，静态成员与实例成员的区别，类与类的分工及交互等等。） | | | | | | |
| **教师批阅** |  | | | | | | |