高级语言程序设计I实验报告

实验四 子类、接口与多态性

班 级： 22软件工程

姓 名： 王玉升

学 号： 22206091012

成 绩：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 实验名称 | | 实验四 子类、接口与多态 | | | |
| 实验日期 | | | 2023.11.6 | 实验课时 | 2 |
| 实  验  目  的  及  要  求 | （给出本次实验内容及要求的具体描述）   1. 实验目的：   1． 掌握子类的继承性及子类对象的创建过程  2．掌握成员变量的继承与隐藏及成员方法的继承与重写  3．掌握上转型对象的使用  4．掌握接口的设计和实现  5．掌握接口回调的使用  6．掌握面向抽象的编程和面向接口的编程   1. 实验要求： 2. 安JDK及Ｅclipse开发环境，并使用Ｅclipse开发平台进行java语言程序的编辑、编译和运行。 3. 在D盘或E盘以自己的学号末尾四位数字为名创建文件夹，并在其下面为每个实验项目创建子文件夹，程序相关文件存储在相应的文件夹中（如学号末尾是”1023”，实验四则应创建目录结构“E:\1023\sy04”，实验四程序文件放在“…\sy04”）。 4. 实验报告中附上程序清单和程序运行结果截图（控制台中输出结果后用自己的真实姓名注释）。 5. 每题程序代码中至少有两处用自己的真实姓名进行注释。   5．实验结束后进行总结。 | | | | |
| 实  验  环  境 | （列出本次实验所使用的平台和相关软件）  地点：南6207  开发环境：JDK1.8、eclipse | | | | |
| 实  验  过  程  及  实  验  结  果 | 1. **银行与利息(了解重写的目的及怎样使用super关键字)请按模板要求，将【代码X】替换为Java程序代码（注释不能删除）：**   **/\*本题重点是掌握子类的定义、继承、成员隐藏、方法重写\*/**   * 1. 程序清单：   **Bank.java**  public class Bank {     int savedMoney; //存款     int year; //存款年限     double interest; //利息     double interestRate = 0.29; //利率     public double computerInterest() { //计算利息        interest=year\*interestRate\*savedMoney;        return interest;     }     public void setInterestRate(double rate) { //修改利率        interestRate = rate;     }  }  **ConstructionBank.java**  public class ConstructionBank extends Bank {     double year;     public double computerInterest() {        super.year=(int)year; //将隐藏year设为year的整数        double r = year-(int)year;        int day=(int)(r\*1000); // 计算零头天数，如year =8.236表示8年236天        double yearInterest = 【代码1】 //【代码1】调用父类隐藏的方法计算整年的利息        double dayInterest = day\*0.0001\*savedMoney;        interest= yearInterest+dayInterest;        System.out.printf("%d元存在建设银行%d年零%d天的利息:%f元\n ", savedMoney,super.year,day,interest);        return interest;     }  }  **BankOfDalian.java**  public class BankOfDalian extends Bank {     double year;     public double computerInterest() {       【代码2】//【代码2】将父类隐藏year设为year的整数        double r = year-(int)year;       【代码3】//【代码3】计算零头天数        double yearInterest = 【代码4】//【代码4】调用父类隐藏的方法计算整年的利息        double dayInterest = day\*0.00012\*savedMoney;       【代码5】//【代码5】计算总利息        System.out.printf("%d元存在大连银行%d年零%d天的利息:%f元\n ",savedMoney,super.year,day,interest);        return interest;     }  }  **SaveMoney.java**  public class SaveMoney {     public static void main(String args[]) {      int amount=8000;      ConstructionBank bank1 = new ConstructionBank();      bank1.savedMoney = amount;      bank1.year = 8.236;//8年236天       bank1.setInterestRate(0.035);      double interest1 = bank1.computerInterest();      BankOfDalian bank2 = new BankOfDalian();      bank2.savedMoney = amount;      bank2.year = 8.236;      bank2.setInterestRate(0.035);      double interest2=bank2.computerInterest();      System.out.printf("两个银行利息相差%f元\n ",interest2-interest1);     }  }   * 1. 运行程序和运行结果（截图）：\      1. **用面向抽象的编程方法改写实验三中第3题，有以下类：**   **Shape：抽象类，有getArea()抽象方法，求面积。**  **Triangle ：三角形类，是Shape子类，有side1、side2、side3三个成员变量，表示三角形三边，有构造方法（有三个参数），并重写了getArea()方法。**  **Lader: 梯形类，是Shape子类，有above、bottom、height三个成员变量，表示梯形上底、下底和高，有构造方法（有三个参数），并重写了getArea()方法。**  **Circle ：圆类，是Shape子类，有r成员变量，表示圆的半径，有构造方法（有一个参数），并重写了getArea()方法。**  **ComputeShape：面向抽象类Shape的类，该类有一个showMess(Shape shape)方法，该方法计算形状的面积，即通过上转型对象调用写的getArea()来获取面积，并显示。**  **Sy4\_2：主类，创建ComputeShape对象，调用showMess(Shape shape)方法，分别显示三角形、梯形和圆的面积**  **编写该程序的代码：**  **/\*本题重点是掌握抽象方法的定义，子类的定义和方法的重写，面向抽象的编程\*/**   |  | | --- | | ComputeShape | |  | | showMess(Shape shape):void |  |  | | --- | | Shape | |  | | getArea():double |  |  | | --- | | Circle | | r : double | | getArea():double |  |  | | --- | | Triangle | | side1: double  side2: double  side3: double | | getArea():double |  |  | | --- | | Lader | | above: double  bottom : double  height: double | | getArea():double |   **图4-1 UML图**  **程序执行结果形式：**  三角形的面积是：6.0  三角形的面积是：24.0  三角形的面积是：28.26   * 1. 程序清单：   public abstract class Shape {      public abstract double getArea();  }  public class Lader extends Shape {  double above;  double bottom;  double height;    public Lader(double a, double b, double c){  above = a;  bottom = b;  height = c;    }    public double getArea() {    return (above + bottom) \* height / 2;  }  }  public class Circle extends Shape {    double r;    public Circle(double a) {  r = a;  }    public double getArea() {    return 0.5 \* 3.14 \* r \* r;  }  }  public class Triangle extends Shape {  double side1;  double side2;  double side3;    public Triangle(double a, double b, double c) {  side1 = a;  side2 = b;  side3 = c;  }    public double getArea() {  return 0.5 \* side1 \* side2;  }  }  public class ComputeShape extends Shape {  public void showMess(Shape shape){  System.out.println(shape.getArea());  }  @Override  public double getArea() {  return 0;  }  }  public class main2 {  public static void main(String[] args) {  ComputeShape Cs = new ComputeShape();  Shape L = new Lader(1.0, 2.0, 3.0);  Cs.showMess(L);  Shape T = new Triangle(3.0, 4.0, 5.0);  Cs.showMess(T);  Shape C = new Circle(3.0);  Cs.showMess(C);  }   * 1. 运行程序和运行结果（截图）：      1. **面向接口的程序设计：下面程序模拟狗在不同情况下的表现：DogState接口是狗表现的抽象，定义有showState（）方法，类SoftlyState、MeetEnemyState、MeetFriendState、MeetAnotherDog实现了DogState接口，模拟狗在不同情况下的不同表现，将【代码X】替换为Java程序代码（每处有可能是多条语句，注释不能删除）：**   **/\*本题重点是掌握接口的定义、接口的实现、接口回调\*/**   * 1. 程序清单：   public interface DogState {  public void showState();//【代码3】重写 showState()方法，狗在朋友面前表现为：晃动尾巴,表示欢迎  }  【代码1】//【代码1】定义名为DogState的接口，接口中只有showState()方法  class SoftlyState implements DogState {  public void showState() { //重写 showState()方法，狗在主人面前表现为：听主人的命令  System.out.println("听主人的命令");  }  }  class MeetEnemyState implements DogState {  @Override  public void showState() {  System.out.println("狂叫，并冲向去很咬敌人");  System.out.println("wys 上前制止");  }  //【代码2】重写 showState()方法，狗在敌人面前表现为：狂叫，并冲向去很咬敌人  }  class MeetFriendState implements DogState {  @Override  public void showState(){  System.out.println("晃动尾巴,表示欢迎");  } //【代码3】重写 showState()方法，狗在朋友面前表现为：晃动尾巴,表示欢迎  class Dog { //面向接口的类  DogState state; //接口成员变量  public void setState(DogState s) {  state = s;  }  public void show(){  state.showState();  }//【代码5】定义方法show(),显示狗的表现，接口回调  }  }【代码5】 //【代码5】定义方法show(),显示狗的表现，接口回调  }  class MeetAnotherDog implements DogState {  @Override  public void showState() {  System.out.println("嬉戏");  }  //【代码4】重写 showState()方法，狗在同伴面前表现为：嬉戏  }  public class CheckDogState {  public static void main(String args[]) {  Dog yellowDog =new Dog();  System.out.print("狗在主人面前:");  yellowDog.setState(new SoftlyState());  yellowDog.show();  System.out.print("狗遇到敌人:");  yellowDog.setState(new MeetEnemyState());//【代码6】显示狗遇到敌人的表现  yellowDog.show();  System.out.print("狗遇到朋友:");  yellowDog.setState(new MeetFriendState());  yellowDog.show();  //【代码7】显示狗遇到朋友的表现  System.out.print("狗遇到同伴:");  yellowDog.setState(new MeetAnotherDog());  yellowDog.show();  //【代码8】显示狗遇到同伴的表现  }  }  （2） 运行程序和运行结果（截图）：     1. **问题的提出**   **定义：现在要开发一个应用，模拟移动存储设备的读写，即计算机与U盘、MP3、移动硬盘等设备进行数据交换。**  **环境：已知要实现U盘、MP3播放器、移动硬盘三种移动存储设备，要求计算机能同这三种设备进行数据交换（读、写、播放），并且以后可能会有新的第三方的移动存储设备，所以计算机必须有扩展性，能与目前未知而以后可能会出现的存储设备进行数据交换。各个存储设备间读、写的实现方法不同，U盘和移动硬盘只有这两个方法，MP3Player还有一个PlayMusic方法。**  **典型解决方案列举：**  **方案一：分别定义FlashDisk、MP3Player、MobileHardDisk三个类，实现各自的Read和Write方法。然后在Computer类中实例化上述三个类，为每个类分别写读、写方法。例如，为FlashDisk写ReadFromFlashDisk、WriteToFlashDisk两个方法。总共六个方法。缺点：（1）各个类独立，不能实现继承。（2）直接面向具体问题细节，不利集中精力进行整体设计。（3）直接面向具体问题，不利于应对问题的变化的功能拓展**  **方案二： 面向抽象的设计，定义抽象类MobileStorage，在里面写虚方法Read和Write方法，三个存储设备继承此抽象类，并重写Read和Write方法。Computer类中包含一个类型为MobileStorage的成员变量，并为其设有读数据和写数据两个方法：ReadData和WriteData，并通过多态性实现不同移动设备的读写。优点：（1）容易实现继承。（2）面向抽象的设计，有利集中精力进行整体设计。（3）面向抽象的设计，有利于应对问题的变化的功能拓展**   |  | | --- | | MobileStorage | |  | | Read():void  Write():void |  |  | | --- | | Computer | | storage :MobileStorage | | set(IMobileStorage) :void  ReadData ():void  Write Data ():void |  |  | | --- | | MobileHardDisk | |  | | Read():void  Write():void |  |  | | --- | | MP3Player | |  | | Read():void  Write():void  PlayMusic():void |      |  | | --- | | FlashDisk | |  | | Read():void  Write():void |   **图4-2** UML图  **方案三：面向接口的设计，与方案二基本相同，只是不定义抽象类，而是定义接口IMobileStorage，移动存储器类实现此接口。Computer中通过依赖接口IMobileStorage实现多态性。优点：（1）面向接口的设计，有利集中精力进行整体设计。（2）面向接口的设计，有利于应对问题的变化的功能拓展**   |  | | --- | | <<interface>>  *IMobileStorage* | |  | | Read():void  Write():void |  |  | | --- | | Computer | | storage :IMobileStorage | | set(IMobileStorage) :void  ReadData ():void  Write Data ():void |  |  | | --- | | MobileHardDisk | |  | | Read():void  Write():void |  |  | | --- | | MP3Player | |  | | Read():void  Write():void  PlayMusic():void |  |  | | --- | | FlashDisk | |  | | Read():void  Write():void |     **图4-3** UML图   |  | | --- | | <<interface>>  *IReadable* | |  | | Read():void |   **方案四：定义接口IReadable和IWritable，两个接口分别只包含Read和Write，然后定义接口IMobileStorage接口继承自IReadable和IWritable，剩下的实现与方案三相同。**   |  | | --- | | Computer | | storage :IMobileStorage | | set(IMobileStorage) :void  ReadData ():void  Write Data ():void |  |  | | --- | | <<interface>>  *IMobileStorage* | |  | | Read():void  Write():void |  |  | | --- | | <<interface>>  *IReadable* | |  | | Write():void |  |  | | --- | | FlashDisk | |  | | Read():void  Write():void |  |  | | --- | | MP3Player | |  | | Read():void  Write():void  PlayMusic():void |  |  | | --- | | MobileHardDisk | |  | | Read():void  Write():void |   **图4-4** UML图  **请按上述方案二进行程序设计**  **/\*本题重点是掌握面向抽象的程序设计、方法重写、多态性及上转型对象的正确使用\*/**  **程序运行结果：**  U盘读完成！  U盘写完成！  移动硬盘读完成  移动硬盘写完成  MP3读完成！  MP3写完成！   * 1. 程序清单：   abstract public class MobileStorage {  abstract public void read();  abstract public void write();  }  public class FlashDisk extends MobileStorage {  @Override  public void read(){  System.out.println("U盘读完成！");  }  @Override  public void write(){  System.out.println("U盘写完成！");  }  }  public class MobileHardDisk extends MobileStorage{  @Override  public void read(){  System.out.println("移动硬盘读完成");  }  @Override  public void write(){  System.out.println("移动硬盘写完成");  }  }  public class MP3Player extends MobileStorage{  @Override  public void read(){  System.out.println("MP3读完成！");  }  @Override  public void write(){  System.out.println("MP3写完成！");  }  public void PlayMusic(){  System.out.println("MUSIC正在播放！");  }  }  public class Computer {  MobileStorage storage;  public void set(MobileStorage storage){  this.storage = storage;  }  public void ReadData(){  storage.read();  }  public void WriteData(){  storage.write();  }  }  public class main4 {  public static void main(String[] args) {  Computer c = new Computer();  c.set(new FlashDisk());  c.WriteData();  c.ReadData();  c.set(new MobileHardDisk());  c.WriteData();  c.ReadData();  c.set(new MP3Player());  c.WriteData();  c.ReadData();  }  }   * 1. 运行程序和运行结果（截图）：      1. **请按上题方案三编程解决此问题。**   **/\*本题重点是掌握面向接口的程序设计、方法重写、多态性及接口回调\*/**   * 1. 程序清单：   2. 运行程序和运行结果（截图）：  1. **完成教材P143第四题的编程。**   **设计一个动物声音“模拟器”，能模拟许多动物的叫声。参照第4题方案二采用面向抽象编程的方法进行程序设计，要求如下：**   * **抽象类Animal：**   **Animal类有两个抽象方法cry()和getAnimalName()，要求各种具体动物给出自己的叫声和种类名称。**   * **模拟器类Simulator**   **该类面向抽象类Animal设计，有一个playSound(Animal animal)方法，该方法的参数是Animal类型，animal可以调用Animal子类重写的cry()方法播放具体动物的声音，调用子类重写的getAnimalName()方法显示动物种类的名称。**   * **Animal的子类：Dog类和Cat类**  |  | | --- | | Animal | |  | | cry():void  getAnimalName():String |  |  | | --- | | Simulator | |  | | playSound(Animal):void |  |  | | --- | | Cat | | name:String | | cry():void  getAnimalName():String |  |  | | --- | | Dog | | name:String | | cry():void  getAnimalName():String |   **图4-5** UML图   * **主类Sy4\_6**   **main方法中实现模拟播放狗和猫的叫声，并显示它们的种类名称**  **/\*本题重点是掌握面向抽象的程序设计、方法重写、多态性及上转型对象的正确使用\*/**  **程序执行结果：**  小狗汪汪叫  名字是:旺财狗  小猫喵喵叫  名字是:发财猫   * 1. 程序清单：   2. 运行程序和运行结果（截图）：  1. **参照第4题方案三采用面向接口编程的方法完成第6题的程序设计。**   **提示如下：**   * **将第6题的Animal抽象类改接口，Dog类和Cat类实现Animal接口，模拟器类Simulator面向接口Animal设计**  |  | | --- | | <<interface>>  *Animal* | |  | | cry():void  getAnimalName():String |  |  | | --- | | Simulator | |  | | playSound(Animal):void |  |  | | --- | | Dog | |  | | cry():void  getAnimalName():String |  |  | | --- | | Cat | |  | | cry():void  getAnimalName():String |   **图4-6** UML图   * **主类Sy4\_7**   **main方法中实现模拟播放狗和猫的叫声，并显示它们的种类名称**  **/\*本题重点是掌握面向接口的程序设计、方法重写、多态性及接口回调\*/**   * 1. 程序清单：   2. 运行程序和运行结果（截图）：  1. **假设一个学校接待方面的程序，招待不同身份的人的食宿问题，其对应规则如下：**     **理论上，当然可以对每个不同身份的人各定义一个对应的类，并实现各自的方法，但是观察这写类，可以归纳出其有一个共同的模板，即“人”的“食、宿”问题。这时候，就可以发挥抽象类或接口的功能了。请同学们参照第4题方案二或方案三，选择之一编写程序，实现如下输出：**  **程序执行结果：**  学生去食堂吃饭！  学生回寝室睡觉！  教师去教工餐厅吃饭！  教师回学校公寓睡觉！  家长去招待所饭馆吃饭！  家长回招待所睡觉！   * 1. 程序清单：   2. 运行程序和运行结果（截图）： | | | | |
| 总  结 | 实验心得体会： | | | | |