实验课程名称：Java语言程序设计A

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **实验项目名称** | **实验3：类与对象** | | | **实验成绩** |  |
| **实 验 者** |  | **专业班级** | **信管15\*\*** | **组 别** | **1人1组** |
| **同 组 者** | **无** | | | **开始日期** | **2018年4月3日** |
| 第一部分：实验预习报告（包括实验目的及意义，实验基本原理与方法，主要仪器设备及耗材，实验内容及要求，实验方案与技术路线等）  一．实验目的及意义  1．根据实际需求，使用UML类图，合理设计类及成员（实例变量、方法）。  2．编写、调用构造函数，初始化新建对象。  3．编写、调用方法，完成相关操作。  4．实现信息隐藏与封装。  5．创建对象、调用对象方法。  6．正确使用对象数组。  7．设计、实现算法逻辑。  二．实验基本原理与方法  1．面向对象的概念、思想、思维方式。  2．UML面向对象设计初步。  2．类的概念、结构、定义、使用。  3．构造函数的概念、结构、定义、调用。  4．方法的概念、结构、定义、调用。  5．信息隐藏与封装的概念、实现。  6．对象的概念、创建、使用。  7．对象数组的概念、使用。  8．控制流程。  三．主要仪器设备及耗材  1．PC及其兼容机。  2．计算机操作系统。  3．Java开发工具JDK。  4．程序编辑器，如EditPlus。  四．实验内容及要求  实验3-1：设计一个Student类，该类的实例变量有学号studentNumber、姓名studentName、  专业specialty、学费tuition。学费tuition实现封装，最低学费4500元。请绘制完整的UML类图，写出类Student的Java源代码。然后，编写测试类StudentTest，并给出运行结果。  本实验应用类的设计、UML类图、信息隐藏与封装、测试类。  实验3-2：编写类Book，其实例变量简化为：  bookName //图书名称  numRated5 //将该图书评级为5星（很好）的人数  numRated4 //将该图书评级为4星（好）的人数  numRated3 //将该图书评级为3星（一般）的人数  numRated2 //将该图书评级为2星（差）的人数  numRated1 //将该图书评级为1星（很差）的人数  Java预定义有Book类，我们这里自定义的Book类，将自动屏蔽系统预定义类。  类Book中的方法addRating( )应保证能接收一个有效整数（1-5之间的整数），将与输入整数匹配的评级人数加1。例如，如果接收3，则numRated3加1。  类Book中的方法getAverage( )，返回所有图书评级的平均值。  编写类BookReview，在main( )方法中，创建一个Book对象，调用该对象的方法addRating( )为对应图书评级6次，最后输出该图书的书名、平均评级。  输出结果示例如下：  Head First Java，Average Rating: 4.3  实验报告中必须绘制Book类、BookReview类的UML类图。  实验3-3：实验3-2的数组版本，使用数组代替变量保存图书评级，所有修改只应局限于类Book内部，类BookReview无需任何修改。  Book类不再使用实例变量numRated5、numRated4、numRated3、numRated2、numRated1，而改使用含有5个元素的int型数组ratings[]。方法addRating( )应保证接收一个有效整数（1-5之间的整数），int型数组ratings[]的索引值代表评级，使用循环，将与参数匹配的评级人数加1。例如：若接收3，则ratings[2]加1。  因为Book类的UML类图发生了改变，所以需重新绘制。  实验3-4（选做）：编写储蓄账户类及测试类，模拟存款，参考以下运行方式：  （1）新开5个储蓄账户，各储蓄账户的编号、户名在新开储蓄账户时初始化，假设每位用户新开储蓄账户时都存入了一笔钱。具体实现时，在储蓄账户测试类中，通过定义对象数组为储蓄账户赋初值。  （2）提示用户从键盘输入储蓄账户编号。  （3）若输入的储蓄账户编号与新开的5个储蓄账户编号之一匹配，则提示用户从键盘输入存款金额，存款金额必须大于0，否则显示错误提示，直到用户输入的存款金额大于0为止。该部分功能通过循环遍历对象数组，寻找与用户输入储蓄账户编号相同的账户编号；通过循环保证用户输入的存款金额合理。  （4）相应储蓄账户的余额被更新。  （5）存款完成后，提示存款成功，显示该储蓄账户的编号、户名、余额。  （6）若输入的储蓄账户编号与新开的5个储蓄账户编号均无法匹配，显示错误提示，并询问是否还要继续存款。该部分功能通过循环实现。  （7）用户输入yes，继续执行步骤（2），否则显示退出信息。  在分析问题的基础上，绘制储蓄账户类、储蓄账户测试类的UML类图。  储蓄账户类中，getAccountNumber()方法返回账户编号，getAccountName()方法返回账户姓名，getBalance()方法返回账户余额，toString()方法返回账户相关信息。  储蓄账户的余额必须确保大于等于0，这需要通过封装实现。  五．实验方案及技术路线（设计思路、UML类图、算法、注意事项）  实验3-1  3-1.1设计思路：   1. 封装学费tuition   {  //为了让tuition更加突出，只对tuition进行private修饰  Get方法  Set方法  }  除了满足实验要求以外，类student再额外定义一个方法输出Student的个人信息，返回值为String形式   1. 编写测试类StudentTest   {  创建一个Student对象，引用变量A指向它；  调用对象内输出信息的方法；  }  3-1.2 UML类图（类Student与类StudentTest为依赖关系）：      实验3-2  3-2.1设计思路：   1. Book类   {  声明五个整型变量，一个字符串变量（numrated1~5，bookname），为减少封装的代码量，变量全为public（默认）；  定义方法addRating，这个方法的返回值是Boolean，它接受一个整数Rank，并做switch-case选择，有break；default情况下，给出错误提示并返回false；无default条件下，返回ture；  定义方法getAverage，使用类中的整型成员进行加权平均值计算，返回这个double型加权平均值；  定义本身构造函数Book（String nameIn），直接确定这本书的名字；  }   1. BookView类   {  需要Scanner对象接收键盘输入流；  创建Book对象；  进入循环体{输入整数评级Rank，调用Book.addRating（Rank），根据Boolean返回值做算法调整}  输出本书信息；  }  3-2.2 UML类图（类Book与BookReview为依赖关系）    实验3-3  3-3.1设计思路：  ① Book类  {  声明一个整型数组，一个字符串变量（numrated[]，bookname），为减少封装的代码量，变量全为public（默认）；  定义方法addRating，这个方法的返回值是Boolean，它接受一个整数Rank，并做switch-case选择，有break；default情况下，给出错误提示并返回false；无default条件下，返回ture；  定义方法getAverage，使用类中的整型成员进行加权平均值计算，返回这个double型加权平均值；  定义本身构造函数Book（String nameIn），直接确定这本书的名字；  }  ② BookReview类  {  需要Scanner对象接收键盘输入流；  创建Book对象；  进入循环体{输入整数评级Rank，调用Book.addRating（Rank），根据Boolean返回值做算法调整}  输出本书信息；  }  3-3.2 UML类图    实验3-4  3-4.1设计思路：  //本实验题目的设计重点在测试类，也就是步骤（1）到（7）的实现，如下伪代码  测试类{   1. 定义三个boolean类型的变量：“输入了正确编号”、“输入了有效的存款数字”、“再来一次”，均先初始化为“FALSE”； 2. 实例化账户类数组[五个]； 3. 进入main函数。设计循环内容，抓住关键字while，如下代码：   while循环（！“再来一次”）  {  ……  while循环（！“输入了正确编号”）{……}  while循环（！“输入了有效的double存款数字”）{……}  ……  }   1. 当用户输入的银行账户编号通过了“输入了正确编号”循环时，boolean变量“输入了正确编号”= TRUE，该变量对应的循环停止；   若不通过，则直接提问：“是否继续输入（Y/N）”；用户不想再输入时（输入了“N”），那么boolean变量“输入了正确编号”和“输入了有效的存款数字”同时为TRUE，变量对应的循环停止。   1. 当用户完成步骤④，并且通过了“输入了有效的double存款数字循环”后，提示：“存款更新成功！”同时输出相应账户的所有信息（在Account类中将toString（）方法重写以达到这个要求），变量“输入了有效的存款数字”=TRUE，该变量对应的循环停止。   若用户乱输变量（数字和小数点以外的字符）导致不通过，循环将强制跳出步骤⑤，到步骤⑥，询问用户是否还要重新执行存款操作。   1. 用户很顺利的完成了步骤④和⑤，“再来一次”变量对应的循环要做提问：“是否要重新存款？（Y/N）”   若用户不想继续存款（“N”），那么“再来一次”=TRUE，大循环终止  若用户还想继续存款（“Y”），那么三个boolean类型的变量全部重置为FALSE，大循环重新开始。  //测试类设计结束  }  账号类的设置不再写明。 | | | | | |
| 第二部分：实验过程记录（可加页）（代码、运行结果、实验中出现的问题及解决过程）  实验3-1：  3-1.1代码部分  **public** **class** StudentTest {  **public** **static** **void** main(String[] args) {  // **TODO** 自动生成的方法存根  Student A = **new** Student("0121503","姓名","信管");  A.setTuition(4677);  System.***out***.println(A.toString());  }  }  **class** Student{  **public** String studentNumber;  **public** String studentName;  **public** String specialty;  **private** **double** tuition;  **public** Student(String stuNumberIn,String stuNameIn,String specialtyIn) {  **this**.studentNumber = stuNumberIn;  **this**.studentName = stuNameIn;  **this**.specialty = specialtyIn;  }  **void** setTuition(**double** tuitionIn) {  **if**(tuitionIn >= 4500.0)  **this**.tuition = tuitionIn;  **else** System.***out***.println("学费应大于等于4500");  }  **double** getTuition() {  **return** tuition;  }  **public** String toString() {  **return** ("姓名：" + **this**.studentName + "\n" + "学号：" + **this**.studentNumber + "\n" + "专业：" + **this**.specialty+ "\n" + "学费：" + **this**.tuition);  }  }  3-1.2运行结果（为了使打印效果更好，减少黑色墨印，没有使用黑色背景的CMD编译）    实验3-2：  3-2.1代码部分  **import** java.util.\*;  **public** **class** BookReview {  **public** **static** **void** main(String[] args) {  Book A = **new** Book("my first book");  **int** i = 0;//计数器，按题目要求6次循环  Scanner keyboard = **new** Scanner(System.***in***);  **while**(i < 6) {  System.***out***.println("第" + (i + 1) + "次输入评级：");  **if**(A.addRating(keyboard.nextInt()))  i = i + 1;  }  System.***out***.println(A.bookName + ",Average Rating:" + String.format("%.2f", A.getAverage()));  }  }  **class** Book{  String bookName;  **int** NumRated5;  **int** NumRated4;  **int** NumRated3;  **int** NumRated2;  **int** NumRated1;  **public** Book(String nameIn) {  **this**.bookName = nameIn;  }  **boolean** addRating(**int** Rank){  **switch**(Rank) {  **case** 1 : NumRated1 = NumRated1 + 1;**break**;  **case** 2 : NumRated2 = NumRated2 + 1;**break**;  **case** 3 : NumRated3 = NumRated3 + 1;**break**;  **case** 4 : NumRated4 = NumRated4 + 1;**break**;  **case** 5 : NumRated5 = NumRated5 + 1;**break**;  **default** : System.***out***.println("评级只有1到5，重新输入吧！");**return** **false**;  }  **return** **true**;  }  **double** getAverage() {  **return** (5 \* NumRated5 + 4 \* NumRated4 + 3 \* NumRated3 + 2 \* NumRated2 + NumRated1)/(NumRated5 + NumRated4 + NumRated3 + NumRated2 + NumRated1);  }  }  3-2.2运行结果：    3-2.3遇见的问题：  在这一个编写中，忘记在main函数声明时加上static，导致编译出错。  属于常见错误，依照编译器改正即可。  实验3-3：  3-3.1代码部分  **import** java.util.\*;  **public** **class** BookReview {  **public** **static** **void** main(String[] args) {  Book A = **new** Book("my first book ");  **int** i = 0;//计数器，按题目要求6次循环  Scanner keyboard = **new** Scanner(System.***in***);  **while**(i < 6) {  System.***out***.println("第" + (i + 1) + "次输入评级：");  **if**(A.addRating(keyboard.nextInt()))  i = i + 1;  }  System.***out***.println(A.bookName + ",Average Rating:" + String.format("%.2f", A.getAverage()));  }  }  **class** Book{  String bookName;  **int**[] ratings =**new** **int**[5];  **public** Book(String nameIn) {  **this**.bookName = nameIn;  }  **boolean** addRating(**int** Rank){  **switch**(Rank) {  **case** 1 : ratings[0] = ratings[0] + 1;**break**;  **case** 2 : ratings[1] = ratings[1] + 1;**break**;  **case** 3 : ratings[2] = ratings[2] + 1;**break**;  **case** 4 : ratings[3] = ratings[3] + 1;**break**;  **case** 5 : ratings[4] = ratings[4] + 1;**break**;  **default** : System.***out***.println("评级只有1到5，重新输入吧！");**return** **false**;  }  **return** **true**;  }  **double** getAverage() {  **return** (5.0 \* ratings[4] + 4 \* ratings[3] + 3 \* ratings[2] + 2 \* ratings[1] + ratings[0])/(ratings[4] + ratings[3] + ratings[2] + ratings[1] + ratings[0]);  }  }  3-3.2运行结果：    3-4.1代码部分  **import** java.util.\*;  **public** **class** accountText {    **static** Account[] *accounts* = **new** Account[5];  **static** Scanner *keyboard* = **new** Scanner(System.***in***);  **public** **static** **void** main(String[] args) {  // **TODO** 自动生成的方法存根  *accounts*[0] = new Account("*0121503*", "*name*\_*one*");  *accounts*[1] = new Account*("0121504*", "*name*\_*two*");  *accounts*[2] = new Account("*0121505*", "*name*\_*three*");  *accounts*[3] = new Account("*0121506*", "*name*\_*four*");  a*ccounts*[4] = new Account("*0121507*", "*name*\_*five*");  **boolean** rightNumberInput = **false**;  **boolean** rightdepositInput = **false**;  **boolean** tryagain = **false**;  String accountNumberInput = "";  **double** balanceReset = 0.0;  String inputYes;  String againYes;  **while**(!tryagain) {  **while**(!rightNumberInput) {  System.***out***.println("从键盘输入储蓄账户编号:");  accountNumberInput = *keyboard*.next();  **for**(Account item : *accounts*) {  **if**(accountNumberInput.equals(item.getAccountNumber())) {  rightNumberInput = **true**;  }  }  **if**(!rightNumberInput) {  System.***out***.println("找不到这个储蓄账户，是否重新输入（Y/N）");  inputYes =*keyboard*.next();  **if**(inputYes.equals("y") || inputYes.equals("Y"))  rightNumberInput = **false**;  System**.*out*.**println**("**程序退出，感谢使用**");**  return**;**  }  }    **while**(!rightdepositInput) {  System.***out***.println("您输入的账号:" + accountNumberInput + ",匹配正确,请更新您的存款(输入金额数字):");  **if**(*keyboard*.hasNextDouble()) {  balanceReset = *keyboard*.nextDouble();  }  **else** {  System.***out***.println("请不要输入数字和小数点以外的文字符号！操作终止");  String a = *keyboard*.next();  rightdepositInput = **true**;  **continue**;  }  **if**(balanceReset > 0.0) {  **for**(Account item : *accounts*) {  **if**(item.getAccountNumber().equals(accountNumberInput)) {  item.setBalance(balanceReset);  System.***out***.println("存款更新成功了！\n" + item.toString());  rightdepositInput = **true**;  }  }  }  **else** {  System.***out***.println("请输入一个大于零的存款余额！");  }  }    System.***out***.println("是否要继续存款？（Y/N）");  againYes = *keyboard*.next();  **if**(againYes.equals("Y")||againYes.equals("y")) {  tryagain = **false**;  rightNumberInput = **false**;  rightdepositInput = **false**;  }  **else**{  System.***out***.println("程序退出，感谢您的使用！");  tryagain = **true**;  }  }  }  }  **class** Account{  **private** String accountNumber;  **private** String accountName;  **private** **double** balance;  **public** **boolean** setBalance(**double** balanceIn) {  **if**(balanceIn > 0) {  balance = balanceIn;  **return** **true**;  }  **else** {  **return** **false**;  }  }  **public** Account(String numberIn,String nameIn){  accountNumber = numberIn;  accountName = nameIn;  }  **public** String getAccountNumber() {  **return** accountNumber;  }  **public** String getAccountName() {  **return** accountName;  }  **public** **double** getBalance() {  **return** balance;  }  **public** String toString() {  **return** ("账号：" + accountNumber + "\n姓名：" + accountName + "\n余额：" + balance);  }  }  3-4.2小程序运行结果（在下一页）： | | | | | |
| 第三部分 结果与讨论（可加页）   1. 实验结果分析   第一，实验3-1中，封装好的变量tuition可以被公有的方法返回，也可以被公有的方法修改，但是绝对不会在其他类中被访问。例如引用变量A不可以访问它指向的Student的tuition成员，“example=A.tuition”是不被允许的。  第二，实验3-2和3-3都要求限制6次输入，于是在设计循环时可以开头while（计数器 < 6）；在循环体内部，调用可返回boolean值的addRating（int Rank）方法，就可以根据方法本身判断计数器是否该加值。  第三，计数器经过适当运算，可直接当做数组下标使用；于是3-2与3-3代码调整量较小。  第四，为了使double类型的值在输出时更加美观（特别是一个经过数学运算后的double数值），预定义类中的公有方法String.format（”%.2f”，double值）可以返回这个保留小数点后两位的字符串类型的数字。  第五，在一个长长的，变量都是int类型的运算式中，若有一个常量写成了double类型（将5写成5.0），那么这个算式结果就会是double类型。  第六，在可选实验题目中，由于业务不停地需要用户输入一个“yesORno”，所以本题目用到了嵌套循环。这里学生的处理是设置Boolean类型的变量将其作为while循环的检查部分，当用户不想继续做某个操作，相应的Boolean变量就会发生改变。  第七，非常重要！字符串的匹配计算不是双等号，而是String.equals（）方法。如果使用双等号比较的只是两个变量的地址而非实际内容！学生因为这个错误花去了较大量的时间查错。   1. 小结、建议及体会   小笔记——public和private，谁是默认值？  这一次实验我们学习了类的封装：对象的私有变量可以通过公有方法返回。同时也绘制了UML的类图，在整个绘制类图的过程中，要注意成员们前面的修饰符号；如果成员前面没有权限修饰符，默认为“包里每一个成员均可访问”。  对于类中的方法成员，它们最好只做一种行为；方法如果进行了某些操作，那么我们在定义时可以设置布尔类型返回值判断这个方法执行的结果。这样处理可以减少我们一部分编码量，if（boolean 方法（））{…}等漂亮的写法也能完成。  第四部分 实验评分标准（教师自行设计）及成绩   |  |  |  |  |  | | --- | --- | --- | --- | --- | |  | 观测点 | 考核目标 | 权重 | 得分 | | 实验预习 | 程序设计方案、技术路线（设计思路、算法、注意事项）科学、合理、可行 | 实验方案设计能力 | 20% |  | | 实验过程 | 1. 按时参加实验，不迟到，不早退 2. 源代码正确、完整、可读性强、编码规范，运行结果正确，实验中出现的问题得到解决 | 学生的实验态度、UML面向对象建模与设计能力、分析并解决问题的专业素养及编程技能、良好的编程风格与习惯 | 50% |  | | 结果分析 | 实验结果分析到位、总结完善、体会深刻 | 学生对实验数据处理与分析的能力；对专业知识的综合应用能力；事实求实的精神 | 30% |  | | 附加分 | 正确、合理地完成选做实验内容 | 克服困难、刻苦钻研的顽强精神与严谨求实的科学作风 | 最高可加20% |  | | 该项实验报告最终得分 | | |  | |   教师签名： | | | | | |