高级语言程序设计I实验报告

**实验五 内部类与异常处理**

班 级： 22软件工程

姓 名： 王玉升

学 号： 22206091012

成 绩：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 实验名称 | | **实验五 内部类与异常处理** | | | |
| 实验日期 | | | 2023.11.13 | 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、了解java内部类的定义与使用  2、掌握java异常捕获与处理  3、掌握java异常抛出声明、异常抛出与自定义异常   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”，实验五则应创建目录结构“E:\1023\sy05”，实验五程序文件放在“…\sy05”）。 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. **统计手机专卖店使用价值2000元内部购物卷销售手机情况，要求用内部类实现，将【代码X】替换为Java程序代码（注释不能删除）：**   **/\*本题重点是掌握内部类的定义与使用\*/**   * 1. 程序清单：   class MobileShop {  InnerPurchaseMoney purchaseMoney1;//用内部类InnerPurchaseMoney声明购物卷对象purchaseMoney1  InnerPurchaseMoney purchaseMoney2; //【代码1】内部类InnerPurchaseMoney声明购物卷对象purchaseMoney2  private int mobileAmount; //手机的数量  MobileShop(){  purchaseMoney1 = new InnerPurchaseMoney (20000);// 创建价值为20000的购物卷purchaseMoney1  purchaseMoney2 = new InnerPurchaseMoney(10000);// 【代码2】创建价值为10000的购物卷purchaseMoney2  }  void setMobileAmount(int m) {  this.mobileAmount = m;//【代码3】设置手机的数据  }  int getMobileAmount() {  return mobileAmount;  }  class InnerPurchaseMoney {//内部类--内部购物卷  int moneyValue; //购物卷金额  InnerPurchaseMoney(int m) {  moneyValue = m;  }  void buyMobile() {  if(moneyValue>=20000) {  mobileAmount -= 6; //2000元内部购物卷买了6部手机  System.out.println("用价值"+moneyValue+"的内部购物卷买了6部手机");  }  else if(moneyValue<20000&&moneyValue>=10000) {  mobileAmount -= 4;//【代码4】 1000元内部购物卷买了3部手机  System.out.println("用价值"+moneyValue+"的内部购物卷买了3部手机");  }  }  }  }  public class NewYear {  public static void main(String args[]) {  MobileShop shop = new MobileShop();  shop.setMobileAmount(30);  System.out.println("手机专卖店目前有" + shop.getMobileAmount() + "部手机");  shop.purchaseMoney1.buyMobile(); //手机专卖店使用价值2000元内部购物卷销售手机  shop.purchaseMoney2.buyMobile(); //【代码5】手机专卖店使用价值1000元内部购物卷销售手机  System.out.println("手机专卖店目前有" + shop.getMobileAmount() + "部手机");  }   * 1. 运行程序和运行结果（截图）：  1. **用匿名类对象作为方法参数，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）：**   **/\*本题重点是掌握匿名类对象的创建\*/**  （1） 程序清单：  nterface SpeakHello {  void speakHello();  }  class KindHello {  public void lookHello(SpeakHello hello) { //接口类型参数  hello.speakHello(); //接口回调  }  }  public class Sy5\_2 {  public static void main(String args[]) {  KindHello kindHello=new KindHello();  kindHello.lookHello(new SpeakHello() {  @Override  public void speakHello() {  System.out.println("wys,你好，吃饭了吗？");  }  });//【代码1】使用实现接口的匿名类对象，输出"中国人习惯问候语：你好,吃饭了吗? "  kindHello.lookHello(new SpeakHello() {  @Override  public void speakHello() {  System.out.println("wys,你好，天气不错");  }  });//【代码2】使用实现接口的匿名类对象，输出"英国人习惯问候语:你好,天气不错"  }  }（2） 运行程序和运行结果（截图）：   1. **理解 finally 子句的作用**   **当一个异常被抛出时，程序的执行流程就不再是连续的了，会跳过某些语句，甚至会由于没有与之匹配的catch 子句而过早地返回，结束程序的运行。为了确保一段代码不管发生什么异常都能被执行，可以使用finally 子句。每个try 语句至少都要有一个与之相配的catch 或finally 子句。从一个方法中返回到调用它的另外一个方法，可以通过return 语句或通过一个没有被捕获的异常，但finally 子句总是在返回前被执行。**  **创建使用finally 子句的程序文件TestFinally.java**  **/\*本题重点是理解异常捕获与处理\*/**   * 1. 程序清单：   public class TestFinally {  static void mathodA() {//方法mathodA抛出异常，并捕获异常  try {  System.out.println("\nmathodA 抛出一个异常");  throw new RuntimeException();  } catch (RuntimeException e) {  System.out.println("捕获 mathodA 的异常 ");  return;  } finally {  System.out.println("执行 mathodA 的 finally");  }  }  static void mathodB() {//【代码1】方法mathodB的方法头，声明抛出Exception异常  System.out.println("mathodB抛出一个异常");  throw new RuntimeException();//抛出异常，未进行捕获  }  static void mathodC() {  try {  System.out.println("mathodC抛出一个异常");  throw new RuntimeException();  } catch (RuntimeException e) {//【代码2】捕获Exception异常  System.out.println("捕获mathodC的异常 ");  System.exit(0);  } finally {  System.out.println("执行 mathodC 的 finally");  }  }  public static void main(String args[]) {  mathodA();  System.out.println("mathodA方法调用结束 ");  mathodC();  ;//【代码3】调用mathodB()方法,并对异常进行捕获处理  System.out.println("mathodB方法调用结束 ");  mathodC();  System.out.println("mathodC方法调用结束 ");  }   * 1. 运行程序和运行结果（截图）：   2. 分析return与System.exit()的不同：   3. 将方法mathodA 中的catch(RuntimeException e)改成catch(ArithmeticException e)，分析输出的结果。  1. **编程解决下面问题**   **（1）编写下面异常类：**  **类SalarySmall：工资低异常，异常对象.** **getMessage ()方法返回值设为"工资不能低于最低工资!"信息**  **类SalaryBig：工资高异常，异常对象.** **getMessage ()方法返回值设为"工资不能高于最高工资!"信息**  **（2）员工类**  **1）属性：编号、姓名、年龄、工资。**  **2）构造方法：有四参数，设置编号、姓名、年龄、工资。如果工资低于1000元（最低工资），抛出工资低异常，如果工资大于50000元（工资上限），抛出工资高异常。构造方法不对异常时行处理**  **3）方法：**  **addSalary(double salary)：增加工资，salary为工资增加量，当增加工资后高于工资上限，抛出工资高异常。**  **minusSalary(double salary)：减少工资，salary为工资减少量，当减少工资后低于最低工资线，抛出工资低异常。**  **show()：显示职工信息**  **(3)主类Sy5\_4:**  **添加员工，测试构造器中各种异常**  **每名员工增加工资和减少工资，测试方法抛出的异常**  **显示每名员工的信息。**  **/\*本题重点是掌握面自定义异常类的定义、异常的抛出、捕获与处理\*/**   * 1. 程序清单：   **SalarySmall.java:**  class SalarySmall extends Exception{//定义工资低异常类  public SalarySmall(){  super("工资不能低于最低工资!"); //调用父类构造方法，异常对象. getMessage ()方法返回值设为"工资不能低于最低工资!"信息  }  }  **SalaryBig.java:**  【代码1】//【代码1】定义工资高异常类，异常对象. getMessage ()方法返回值设为"  **Employee.java:**  class Employee {  long number;  String name;  int age;  double salary;  【代码2】{//【代码2】构造方法的方法头  this.number=number;  this.name=name;  this.age=age;  if(salary<1000)  throw new SalarySmall();  else if(salary>50000)  【代码3】//【代码3】抛出工资高异常类  else this.salary=salary;  }  【代码4】//【代码4】定义addSalary(double salary)方法，参考构造方法写，如果有异常抛出，如果无异常，增加工资，并输出“工资增加…元”  【代码5】//【代码5】定义minusSalary(double salary)方法，参考构造方法写，如果有异常抛出，如果无异常，减少工资，并输出“工资减少…元”  void show(){  System.out.println("编号:"+number+",姓名:"+name+",年龄:"+age+",工资:"+salary);  }  }  **Sy5\_4.java：**  public class Sy5\_4{  static Employee employee1;  public static void main(String args[]){  try{//创建属性为(1001, "张三",27,500)的员工，并各模拟一次减少500元工资，增加1000元工资  employee1=new Employee(1001, "张三",27,500);  employee1.minusSalary(500);  employee1.addSalary(1000);  employee1.show();  }catch(Exception e){  System.out.println("张三,"+e. getMessage ());  }  try{ //创建属性为(1002,"李四",40,60000)的员工，并各模拟一次减少500元工资，增加1000元工资  employee1=new Employee(1002,"李四",40,60000);  employee1.minusSalary(500);  employee1.addSalary(1000);  employee1.show();  }catch(Exception e){  System.out.println("李四,"+e. getMessage ());  }  【代码6】//【代码6】创建属性为(1003,"王五",30,1500)的员工，并各模拟一次减少1000元工资，增加1000元工资，模仿employee1员工的形式写    【代码7】 //【代码7】创建属性为(1003,"陈虹",45,47000)的员工，并各模拟一次减少1000元工资，增加5000元工资，模仿employee1员工的形式写    try{ //创建属性为(1005,"汪成功",45,40000)的员工  employee1=new Employee(1005,"汪成功",45,40000);  employee1.minusSalary(1000);  employee1.addSalary(5000);  employee1.show();  }catch(Exception e){  System.out.println("陈虹,"+e. getMessage ());  }  }  }   * 1. 运行程序和运行结果（截图）： | | | | |
| 总  结 | 实验心得体会： | | | | |