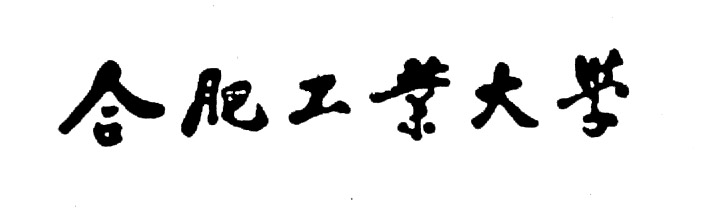
****



**《Java技术》实验报告**

|  |  |
| --- | --- |
| **学院名称：** | **计算机与信息学院** |
| **专业班级：** | **计算机 21级 1班** |
| **姓 名：** | **杨程锦** |
| **学　 号：** | **2021214710** |
| **电子邮箱：** |  |
| **提交时间：** | **2022年7月5日** |

**《Java技术》实验报告**

**实验一： 2022年6月 25 日**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **学院** | 计算机与信息学院 | | **专业班级** | 计算机科学与技术21-1班 | **姓名** | **杨程锦** | **成绩** |  |
| **课程**  **名称** | Java技术 | **实验项目**  **名 称** | | 实验一 Java开发环境的安装与配置 | | | **指导 教师** | 路强 |
| **一、实验目的**   * 1. 掌握JDK的安装步骤（有条件的话可以在Linux系统下安装）。   2. 理解环境变量PATH, CLASSPATH的作用，以及它们的设置方法(Win98和Win2000/XP、Win7、Win8下的设置方法有所区别)。   3. 熟悉记事本（或JCreator）编辑环境，编写简单的Application程序，并编译和解释执行。   4. 记录编译和执行Java程序当中的系统错误信息提示，并给出解决办法。   5. 复习、理解OOP的三大特征——封装、继承和多态。   6. 学会使用Java编写简单的类。理解类的属性和方法。   7. 掌握关键字private, protected, public的作用，掌握静态类，内部类的概念。   8. 掌握多态的概念。   **二、实验原理**  1．掌握安装  2. 学会配置环境  3. 掌握基本的编写Java的方法  **三、使用硬件、软件环境**  PC 计算机一台  处理器 AMD Ryzen 7 5800H with Radeon Graphics 3.20 GHz  机带 RAM 32.0 GB (31.9 GB 可用)  系统类型 64 位操作系统, 基于 x64 的处理器  安装Win10操作系统。另外，使用IDEA，JDK8等软件  **四、实验过程、步骤及原始记录(算法、原程序、测试结果，分析等)**  1. 下载最新版本的JDK，并安装。  2. 设置环境变量PATH, CLASSPATH, 使得Java程序能正确编译和执行。  3. 在记事本（或JCreator）环境下编写一个HelloWorld.java程序，在DOS提示符下编译并执行这个程序。    4. 编写一个类A，它位于包a中，保存到A.java中，再编写一个类B，它位于包b中，保存到B.java中。在类B中实例化类A的一个对象。分别编译类A和类B。  A.java  package com.A;  public class A {  public A(){  }  }  B.java  package com.B;  import com.A.\*;  public class B {  public static void main(String[] args) {  A first = new A();  }  }    5. 分别编写两个类Point2D，Point3D来表示二维空间和三维空间的点，使之满足下列要求：  1) Point2D有两个整型成员变量x, y (分别为二维空间的X,Y方向坐标)，Point2D的构造方法要实现对其成员变量x, y的初始化。  2) Point2D有一个void型成员方法offset(int a, int b)，它可以实现Point2D的平移。  3) Point3D是Point2D的直接子类，它有有三个整型成员变量x,y,z (分别为三维空间的X,Y,Z方向坐标)，Point3D有两个构造方法：Point3D(int x, int y, int z)和Point3D(Point2D p, int z)，两者均可实现对Point3D的成员变量x, y, z的初始化。  4) Point3D有一个void型成员方法offset(int a, int b, int c)，该方法可以实现Point3D的平移。  5) 在Point3D中的主函数main()中实例化两个Point2D的对象p2d1，p2d2，打印出它们之间的距离，再实例化两个Point3D的对象p3d1，p3d2，打印出他们之间的距离。  Point2D.java  public class Point2D {  int x, y;  public Point2D(int x, int y){  this.x = x;  this.y = y;  }   public Point2D() {  }   public void offset(int a, int b){  this.x = a;  this.y = b;  } }  Point3D.java  import java.math.\*;  public class Point3D extends Point2D {  int z;  public Point3D(int x, int y, int z) {  super(x, y);  this.z = z;  }  public Point3D(Point2D p, int z) {  this.x = p.x;  this.y = p.y;  this.z = z;  }  public void offset(int a, int b, int c) {  this.x = a;  this.y = b;  this.z = c;  }  public static void main(String[] args) {  Point2D p2d1 = new Point2D(10, 20);  Point2D p2d2 = new Point2D(20, 40);  double p2distance = Math.sqrt(Math.pow((p2d1.x - p2d2.x), 2) + Math.pow((p2d1.y - p2d2.y), 2));  System.out.printf(String.valueOf(p2distance));  System.out.println(" ");  Point3D p3d1 = new Point3D(p2d1, 30);  Point3D p3d2 = new Point3D(p2d2, 40);  double p3distace = Math.sqrt(Math.pow((p3d1.x - p3d2.x), 2) + Math.pow((p3d1.y - p3d2.y), 2) + Math.pow((p3d1.z - p3d2.z), 2));  System.out.println(String.valueOf(p3distace));  }  }    分析：  通过编写Point2D类进行点的初始化，再在3D类中通过继承实例化具有三个坐标的点，再在主函数中通过Math类的静态方法sqrt和pow利用勾股定理进行计算两点之间距离。  **五、实验结论、分析、思考题与心得体会**  通过本次实验，我有以下几方面的收获、认识和体会：  1.学会了在Java环境下编写程序及运行程序的方法；  2.加深了对继承和多态的理解  3.学到了关于Java虚拟机的知识，初步理解了Java内存管理  思考题：  1.什么是JVM，它的作用是什么？  JVM是Java Virtual Machine（Java虚拟机）的缩写，JVM是一种用于计算设备的规范，它是一个虚构出来的计算机，是通过在实际的计算机上仿真模拟各种计算机功能来实现的。  引入Java语言虚拟机后，Java语言在不同平台上运行时不需要重新编译。Java语言使用Java虚拟机屏蔽了与具体平台相关的信息，使得Java语言编译程序只需生成在Java虚拟机上运行的目标代码（字节码），就可以在多种平台上不加修改地运行。  JVM内存布局规定了Java在运行过程中内存申请、分配、管理的策略，保证了JVM高效稳定运行。不同JVM对于内存的划分方式和管理机制存在差异  内存管理：  （1）对于Java运行时涉及到的存储区域主要包括程序计数器、Java虚拟机栈、本地方法栈、java堆、方法区以及直接内存等等。对于每个部分，都有其使用的条件。程序计数器主要是取下一条指令，在Java里面主要是取下一条指令的字节码文件；Java虚拟机栈主要是利用栈先进后出的特性存储局部变量表，动态链接等，主要包括堆内存和栈内存，对于程序员内存分析而言是特别重要的。本地方法栈与上边的栈基本作用差不多，只不过这里是为Java方法而服务。Java堆是内存管理中最大的一块，所有的线程共享这一块内容，同时该部分也是垃圾收集器的主要区域。  （2）虚拟机的垃圾回收机制是完善的，动态内存分配和回收是比较成熟的，在内存管理机制中，大部分都不需要我们考虑内存回收，只有Java堆和方法区需要我们考虑处理内存问题。一般的对于内存回收首先就是判断某一个部分是生存还是死亡，主要是通过下面二种算法：  其一是引用计数算法，本算法实现简单，判定的效率也是比较高的，很多的软件都使用了该算法，但是主流的Java并没有选择该算法，核心的问题是该算法难以处理对象之间相互调用的问题。  其二是称可达性分析算法，该算法核心思想是依靠判断对象是否存活来实现的，本算法是通过一系列的GC ROOTS的对象作为起始点，采用搜索的算法遍历引用链，如果搜索过程中没有发现该节点，则认为该节点是不可达的，即可回收的，在Java里面，一般可以使用该算法处理问题。  碎片回收：  Java类的实例所需的存储空间是在堆上分配的。解释器具体承担为类实例分配空间的工作。解释器在为一个实例分配完存储空间后，便开始记录对该实例所占用的内存区域的使用。一旦对象使用完毕，便将其回收到堆中。在Java语言中，除了new语句外没有其他方法为一对象申请和释放内存。对内存进行释放和回收的工作是由Java运行系统承担的。这允许Java运行系统的设计者自己决定碎片回收的方法。在SUN公司开发的Java解释器和Hot Java环境中，碎片回收用后台线程的方式来执行。这不但为运行系统提供了良好的性能，而且使程序设计人员摆脱了自己控制内存使用的风险。  2如何理解PATH和CLASSPATH,，如何设置它们。  classpath环境变量告诉类装载器到哪里去寻找第三方提供的类和用户定义的类。JVM和其他JDK工具通过依次搜索平台库，扩展库,和类路径来查找类。  path环境变量指定了JDK命令搜索路径,设置path的作用是让操作系统可以找到JDK命令。path环境变量原来Windows里面就有，只需修改一下，使他指向JDK的bin目录，这样在控制台下面编译、执行程序时就不需要再键入一大串路径了。设置方法是保留原来的path的内容，并在其后加上%JAVA\_HOME%\bin (注:%%引起来的意思是引用上一步设定好的环境变量JAVA\_HOME，写成D:\JDK\_1.4.2也是可以的。)  设置Classpath  Java执行环境本身就是一个平台，执行这个平台上的程序是已编译完成 的Java程序（Java程序编译完成之后，会以.class文件存在）。如果将Java执行环境比喻为操作系统，如果设置Path变量是为了让操作系统找到指定的工具程序（以Windows来说就是找到了.exe文件），则设置Classpath的目的就是让Java执行环境找到指定的Java程序（也就是.class文件）。  有几个方法可以设置Classpath，最简单的方法是在系统变量中新增Classpath环境变量。在“系统变量”下单击“新建”按钮，在“变量名”文本框中输入Classpath，在“变量值”文本框中输入Java类文件的位置。例如可以输入.;E:\Java9\lib\tools.jar;E:\Java9\lib\rt.jar（jar文件是zip压缩格式，其中包括.class文件和jar中的Classpath设置），每一路径中间必须以“；”作为分隔。第一个用“.；”作为分隔。  总而言之，设置Classpath的目的在于告诉Java执行环境，在哪些目录下可以找到你所要执行的Java程序。一个分辨Path与Classpath的方式就是：“对于Windows”操作系统来说，path是让操作系统可以找到.exe执行程序的存在，而对于Java执行环境来说，Classpath就是让JVM可以找到.class执行程序 的存在“  设置Path  必须告诉操作系统，应该到哪些目录下尝试找到你所想使用的工具程序。有几个方法可以进行这项设置，最简便的方法就是设置系统变量中的Path的环境变量。在"我的电脑"图标上选择并单击右键选择“属性”后切换到“高级”页面，并单击下方的“环境变量”按钮，在“环境变量”对话框中编辑Path变量。在Path变量的“编辑系统变量”对话框中，在“变量值”文本框的最前方输入JDK bin目录的路径，然后紧跟着一个分号，以作为分隔，单击“确定”按钮即可完成设置。（在Win10系统中有一点点不同，此时是将JDK bin的目录的路径像一个条目一样加在已有路径的下方，路径可以上下移动）  之所以要将JDK的路径设置放置在Path变量设置的最前方，是因为操作系统在搜索Path路径设置时，会从最前方开始读取，如果在路径下找到指定程序，就会直接执行。当你的系统中安装在两个以上的JDK时，在Path路径中的设置顺序，将决定执行哪个JDK下的工具程序。  设置Path变量之后，要重新打开一个命令模式才能重新读入Path变量内容，接着如果执行javac程序，应该可以看到与之前不同的画面。（这个是成功的画面）  如果Path设置是寻找JDK安装目录下的bin目录，则操作系统会搜索JDK安装目录下的工具程序。这样当执行java程序时，就会使用JDK附带的JRE，而不是公用的JRE。这两个JRE的安装路径并不相同。  Java中子类对象的创建过程的详细步骤是什么？  1.加载父类class,为父类静态属性分配内存  2.加载子类class,为子类静态属性分配内存  3.为父类静态属性赋值,执行父类静态代码块  4.为子类静态属性赋值,执行子类静态代码块  5.为父类成员属性分配内存  6.为子类成员属性分配内存  7.为父类成员属性赋值  8.执行父类构造方法  9.为子类成员属性赋值  10.执行子类构造方法  4.什么是面向对象的多态性，在程序中如何体现的？  理解多态：可以理解为一个事物的多种形态  何为多态性：  ​ 对象的多态性： 父类的引用指向子类的对象（或 子类的对象赋给父类的引用）  多态的使用，虚拟方法调用  ​ 有了对象的多态性以后，我们在编译期，只能调用父类中声明的方法，但在运行期我们实际执行的是子类重写父类的方法  总结： 编译看左边，运行看右边  多态性的使用前提  1.类的继承关系  2.方法重写 | | | | | | | | |

**《Java技术》实验报告**

**实验二： 2022年6月 30日**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **学院** | 计算机与信息学院 | | **专业班级** | 计算机科学与技术21-1班 | **姓名** | **杨程锦** | **成绩** |  |
| **课程**  **名称** | Java技术 | **实验项目**  **名 称** | | 实验二 货物进销管理系统 | | | **指导 教师** | 路强 |
| 1. **实验目的**   1．掌握Java中文件的读写操作。  2．学会使用Java提供的实用类（Vector, ArrayList）来完成特定的功能。  3．掌握字符串类（String, StringBuffer）的使用。  4．掌握用面向对象的方法分析和解决复杂问题。  **二、实验原理**  通过文件读取的系统类将文件中的信息读入输入流中，再将其读入ArrayList这个容器中，对其进行增删改查，然后将不同情况下对文件不同的处理情况在ArrayList中进行修改，最后再写入输出流中，记入文件中。  **三、使用硬件、软件环境**  PC 计算机一台  处理器 AMD Ryzen 7 5800H with Radeon Graphics 3.20 GHz  机带 RAM 32.0 GB (31.9 GB 可用)  系统类型 64 位操作系统, 基于 x64 的处理器  安装Win10操作系统。另外，使用IDEA，JDK8等软件   1. **实验过程、步骤及原始记录(算法、原程序、测试结果，分析等)**   编写一个Inventory.java完成以下功能：  1．程序首先打开并读取Inventory.txt中记录的所有库存记录，然后读取Transactions.txt，处理这个文件中包含的事务，记录发货记录到Shipping.txt，并记录错误信息到Errors.txt中。最后更新库存到另外一个文件NewInventory.txt中。  2．文件Inventory.txt和NewInventory.txt的每行包含一个存货记录，没条记录包含下面一些字段息，这些字段之间用一个tab分开（见后面的文件格式）：     |  |  | | --- | --- | | 字段 | 格式和含义 | | Item number | 字符串型，货物编号 | | Quantity | 整型，货物数量 | | Supplier | 字符串型，供应商编号 | | Description | 字符串型，货物描述 |   3．字段Items按照从小到大的顺序写入文件的。注意Item号不必连续，如Item号为752的后面可能是800。  4．文件Transactions.txt包含几个不同的处理记录（每行一条记录）。每条记录前面以一个大写字母开头，表示这条记录是什么类型的事务。在不同的大写字母后面是不同的信息格式。所有的字段也是以tab键分开的（见Transactions.txt文件格式）。  5．以'O'开头的事务表示这是一个发货订单，即某一种货物应该发给特定的客户。Item number和Quantity的格式如上面表格定义。Custom编号和上面的Supplier编号一致。处理一条定单记录（以'O'开头的事务）意味着从减少库存记录中相应货物的数量（减少的数量=发货单中的数量），记录发货信息到Shipping.txt中。注意：Inventory.txt中的quantity不应该小于0，如果对于某一种货物，库存的数量小于发货单的数量的话，系统应该停止处理发货单，并记录出错信息到Errors.txt。如果对于某一种货物有多个发货单，而且库存总量小于这些发货单的总和的话，系统应该按照发货单中的数量从小到大的有限原则满足客户。也就是说，对于某一种货物如果一个数量Quantity少的发货单没有处理之前，数量Quantity多的发货单永远不会被处理。（这种处理原则不受发货单记录在Transactions.txt的先后顺序影响）  6．以'R'开头的事务表示这是一个到货单记录，在'R'后面是Item number和它的数量Quanlity。处理一条到货单意味着增加库存中相应货物的数量（增加的数量=到货单中的数量）。注意：如果在Transactions.txt文件中，到货单出现在发货单之后，到货单中的货物数量可以用来填补发货单中的数量（可以理解成在Transactions.txt中，优先处理到货单）。  7．以'A'开头的事务表示向库存中增加一种新的货物（即这种货物以前库存中没有），在'A'后面是Item number，供应商supplier以及货物的描述description。处理一个新增货物记录意味着向库存中增加一个数量Quantity为0的新的Item。你可以假设在一个Transactions.txt中，新增货物记录总是出现在第一个到货单之前。  8．以'D'开头的事务表示从库存中删除一种货物，在'D'后面是Item号。删除操作总是在所有的事物处理之后才被处理，以保证对于可能出现的同一种货物的发货单的操作能在删除之前被正确处理。如果要删除的某种货物的库存量Quantity不为0的话，系统应该向Errors.txt记录出错信息。  9．文件Shipping.txt中的每一行代表给某一客户的发货信息。Shipping.txt中的每一行分别是客户编号、Item号、货物数量，它们之间用tab键分隔。如果发货单中有两条客户编号和Item编号一样的记录，在Shipping.txt中应该将这两条发货信息合并（即将它们的数量相加）。  10．Errors.txt文件包含未发送的发货记录和库存量大于0的删除记录。Errors.txt每一行包含Custom编号、Item编号以及发货单上的数量Quantity。对于删除操作，Custom编号为0，数量Quntity为库存中的Quantity.  11．实验测试数据:  Inventory.txt    Transactions.txt    Inventory.java  import java.io.\*;  import java.util.ArrayList;  import java.util.Collections;  import java.util.Comparator;  public class Inventory {  public static void main(String[] args) throws IOException {  new getInventory();  System.out.println("File have finished!");  }  }  class Goods {  String itemNumber;  int quantity;  String supplier;  String description;  //重载构造函数使其适配两种不同输入质量的构造函数  public Goods (String itemNumber, String quantity, String supplier, String description){  this.description = description;  this.itemNumber = itemNumber;  this.supplier = supplier;  this.quantity = Integer.parseInt(quantity);  }  public Goods (String itemNumber, int quantity, String supplier, String description){  this.description = description;  this.itemNumber = itemNumber;  this.supplier = supplier;  this.quantity = quantity;  }  //获取属性  public String getItemNumber(){  return itemNumber;  }  public int getQuantity(){  return quantity;  }  public String getSupplier(){  return supplier;  }  public String getDescription(){  return description;  }  //修改属性  public void setItemNumber(String itemNumber){  this.itemNumber = itemNumber;  }  public void setQuantity(int quantity){  this.quantity = quantity;  }  public void setSupplier(String supplier){  this.supplier = supplier;  }  public void setDescription(String description){  this.description = description;  }  }  class getInventory {  ArrayList<Goods> goodsArrayList = new ArrayList<Goods>();  BufferedWriter errorWriter = new BufferedWriter(new FileWriter("D:\\Progress\\java\_pratice2\\Document\\Error.txt", true)); //使得字节写入文件末尾而非开头  Goods good = null;  public getInventory() throws IOException {  BufferedReader br = new BufferedReader(new FileReader("D:\\Progress\\java\_pratice2\\Document\\Inventory.txt"));  String line = null;  while ((line = br.readLine()) != null) {  String[] str = line.split("\\s+"); //正则表达式，使得不同字符直接去掉空格，安装内容存入数组  good = new Goods(str[0], str[1], str[2], str[3]);  goodsArrayList.add(good);  }  br.close();  new getTransactions();  }  class getTransactions {  ArrayList<String> transInform = new ArrayList<String>();  ArrayList<String[]> splitedArrayList = new ArrayList<String[]>();  ArrayList<String> shippingBuffer = new ArrayList<String>();  String line = null;  public getTransactions() throws IOException{  BufferedReader readTrans = new BufferedReader(new FileReader("D:\\Progress\\java\_pratice2\\Document\\Transaction.txt"));  while ((line = readTrans.readLine()) != null){ //把文件中内容直接按行读入  transInform.add(line);  }  readTrans.close();  //比较货物操作的优先级  // D < R < O < A  Collections.sort(transInform, new Comparator<String>() {  @Override  public int compare(String o1, String o2) {  if (!(o1.charAt(0) == o2.charAt(0))){  if (o1.charAt(0) == 'O'){  if (o2.charAt(0) == 'A'){  return 1;  } else if (o2.charAt(0) == 'R'){  return -1;  } else if (o2.charAt(0) == 'D'){  return -1;  }  } else if (o1.charAt(0) == 'A'){  return -1;  } else if (o1.charAt(0) == 'D'){  return 2;  } else if (o1.charAt(0) == 'R'){  if (o2.charAt(0) == 'O'){  return 1;  } else if (o2.charAt(0) == 'D'){  return -1;  } else if (o2.charAt(0) == 'A'){  return 1;  }  }  } else {  return (o1.compareTo(o2));  }  return 0;  }  });  //foreach 循环temp容纳容器内每个元素直到没有元素  for (String temp : transInform){  String[] revtal2 = temp.split("\\s+");  splitedArrayList.add(revtal2); //将按行读取的文件去除tab  }  errorFileInfo();  choose();  WriteFile();  }  void errorFileInfo() throws IOException{  errorWriter.write("库存不足信息以及删除货物信息：");  errorWriter.newLine();  errorWriter.write("数字分别代表：客户编号、货物编号、货物数量");  errorWriter.newLine();  }  void choose() throws IOException {  for (String[] temp1 : splitedArrayList){ //遍历每一个操作  switch (temp1[0]){  case "A" :  goodsAdd(temp1);  break;  case "O" :  goodsOutput(temp1);  checkShipping();  break;  case "R" :  goodsReset(temp1);  break;  case "D" :  goodsDelete(temp1);  break;  }  }  }  //发货函数  void goodsOutput(String[] str) throws IOException {  for (Goods good : goodsArrayList){  if (str[1].equals(good.getItemNumber())){  int num = Integer.parseInt(str[2]);  int qual = good.getQuantity();  if (qual > num){  good.setQuantity(qual - num);  String line = good.getSupplier() + "\t" + good.getItemNumber() + "\t" + str[2];  shippingBuffer.add(line);  } else {  String newLine1 = good.getSupplier() + "\t" + good.getItemNumber() + "\t" + str[2];  errorWriter.write(newLine1);  errorWriter.flush();  errorWriter.newLine();  }  }  }  }  //添加货物函数  void goodsAdd(String[] str){  Goods newGood = new Goods(str[1], 0, str[2], str[3]);  goodsArrayList.add(newGood);  }  //到货函数  void goodsReset(String[] str){  for (Goods good : goodsArrayList){  if (str[1] == good.getItemNumber()){  good.setQuantity(good.getQuantity() + Integer.parseInt(str[2]));  }  }  }  //删除货物函数  void goodsDelete(String[] str) throws IOException {  for (Goods good : goodsArrayList){  if (str[1] == good.getItemNumber()){  goodsArrayList.remove(good);  boolean flag = goodsArrayList.remove(good);  if (flag && (good.getQuantity() > 0)){  String line = good.getSupplier() + "\t" + good.getItemNumber() + "\t" + Integer.toString(good.getQuantity());  errorWriter.write(line);  errorWriter.flush();  errorWriter.newLine();  }  }  }  }  //写入NewInventory  void WriteFile() throws IOException {  BufferedWriter bw = new BufferedWriter(new FileWriter("D:\\Progress\\java\_pratice2\\Document\\NewInventory.txt"));  for (Goods newGood : goodsArrayList){  String line = newGood.getItemNumber() + "\t" + Integer.toString(newGood.getQuantity()) + "\t" + newGood.getSupplier() + "\t" + newGood.getDescription();  bw.write(line);  bw.newLine();  bw.flush();  }  bw.close();  }  void checkShipping() throws IOException {  BufferedWriter shippingWriter = new BufferedWriter(new FileWriter("D:\\Progress\\java\_pratice2\\Document\\Shipping.txt"));  for (int i = 0; i < shippingBuffer.size() - 1; i++){  for (int j = i + 1; j < shippingBuffer.size(); j++){  //判断顾客编号和货物编号相同  if ((shippingBuffer.get(i).toString().split("\t")[0].equals(shippingBuffer.get(j).toString().split("\t")[0])) &&  (shippingBuffer.get(i).toString().split("\t")[2].equals(shippingBuffer.get(j).toString().split("\t")[2]))){  int num1 = Integer.parseInt(shippingBuffer.get(i).toString().split("\t")[2]);  int num2 = Integer.parseInt(shippingBuffer.get(j).toString().split("\t")[2]);  int num = num1 + num2;  String line = shippingBuffer.get(i).toString().split("\t")[0] + "\t"  + shippingBuffer.get(i).toString().split("\t")[1] + "\t" + Integer.toString(num);  shippingBuffer.remove(i);  shippingBuffer.remove(j);  shippingBuffer.add(i, line);  i--;  }  }  }  for (String ss: shippingBuffer){  shippingWriter.write(ss);  shippingWriter.newLine();  shippingWriter.flush();  }  shippingWriter.close();  }  }  }  class shortageException extends Exception{  public shortageException(){  super("库存不足");  }  }  class checkOutput{  public static void checkOutput(int num) throws shortageException {  if (num > 0){  System.out.println("数据无误");  } else {  throw new shortageException();  }  }  }    分析：  首先对输入流中的货物操作进行排序，按照增加货物、出货、进货、删除的顺序排序，对同样的操作按照数量进行从小到大排序。对于排序过后是在主函数中进行switch语句的选择，按照顺序进行操作，如果货物不足等等错误则输出在Error文本中，如果一切正常则在输出流中对字符进行操作然后输出在文本中。  **五、实验结论、分析、思考题与心得体会**  通过本次实验，我有以下几方面的收获、认识和体会：  1.对于Java系统类的认识更深一步  2.对于JDK手册的使用更加的熟练  3.对于IDEA的debug功能更加的熟练了 | | | | | | | | |

**《Java技术》实验报告**

**实验三： 2022年7月2日**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **学院** | 计算机与信息学院 | | **专业班级** | 计算机科学与技术21-1班 | **姓名** | **杨程锦** | **成绩** |  |
| **课程**  **名称** | Java技术 | **实验项目**  **名 称** | | 实验三 基于GUI的网络通信程序设计 | | | **指导 教师** | 路强 |
| 教师评语 | 教师签名：  年 月 日 | | | | | | | |
| **一、实验目的**  1．掌握Java中GUI程序的编写，包括事件监听机制。  2．掌握Java的网络通信编程，ServerSocket，Socket类的使用。  3．掌握Java中多线程的编程，Thread类，Runnable接口的使用。  4．掌握用面向对象的方法分析和解决复杂问题。  **二、实验原理**  首先，通过Java中对GUI节目的编写，主要进行了对事件监听器的函数的重写。其次，通过利用ServerSocker和Socket类进行服务器端和客户端的通信，再利用Thread类进行信息的收发同时进行，使其通信过程不间断。  **三、使用硬件、软件环境**  PC 计算机一台  处理器 AMD Ryzen 7 5800H with Radeon Graphics 3.20 GHz  机带 RAM 32.0 GB (31.9 GB 可用)  系统类型 64 位操作系统, 基于 x64 的处理器  安装Win10操作系统。另外，使用IDEA，JDK8等软件  **四、实验过程、步骤及原始记录(算法、原程序、测试结果，分析等)**  编写程序完成以下功能：   1. 设计一个基于GUI的客户-服务器的通信应用程序，如图1，图2所示。  |  |  | | --- | --- | |  |  | | 图1 Socket通信服务器端界面 | 图2 Socket通信客户端界面 |   2.图1为Socket通信服务器端界面，点击该界面中的【Start】按钮，启动服务器监听服务（在图1界面中间的多行文本区域显示“Server starting…”字样）。图2为Socket通信客户端界面，点击该界面中的【Connect】按钮与服务器建立链接，并在图2所示界面中间的多行文本区域显示“Connect to server…”字样，当服务器端监听到客户端的连接后，在图1界面中间的多行文本区域追加一行“Client connected…”字样，并与客户端建立Socket连接。  3.当图1所示的服务器端和图2所示的客户机端建立Socket连接后，编程实现服务端、客户端之间的“单向通信”：在客户端的输入界面发送消息，在服务端接收该消息，并将接收到对方的数据追加显示在多行文本框中。  Socket.java  package pratice3;  import java.awt.\*;  import java.awt.event.ActionEvent;  import java.awt.event.ActionListener;  import java.awt.event.WindowAdapter;  import java.awt.event.WindowEvent;  import java.io.BufferedReader;  import java.io.IOException;  import java.io.InputStreamReader;  import java.io.PrintWriter;  import java.net.Socket;  class myClientFrame extends Frame{  Socket client = null;  TextField port = new TextField("8888");  TextField ip = new TextField("127.0.0.1");  Button connect = new Button("Connect");  TextArea TA = new TextArea(25, 60);  TextField TF = new TextField(40);  Button send = new Button("Send");  myClientFrame(String title){  super(title);  init();  connect.addActionListener(new ActionListener() {  @Override  public void actionPerformed(ActionEvent e) {  try {  connect.setEnabled(false);  int i = Integer.parseInt(port.getText());  TA.append("Socket Connect IP:" + ip.getText() + "\nConnect Port:" + port.getText() + "\n");  client = new Socket(ip.getText(), i);  TA.append("Connect...\n");  ClientThread CT = new ClientThread();  CT.start();  } catch (IOException ex) {  ex.printStackTrace();  }  }  });  send.addActionListener(new ActionListener() {  @Override  public void actionPerformed(ActionEvent e) {  try {  String ss = TF.getText();  PrintWriter aa = new PrintWriter(client.getOutputStream());  if (ss != ""){  aa.write("Client send:" + ss + "\n");  TA.append("Client send:" + ss + "\n");  aa.flush();;  }  TF.setText("");  } catch (IOException ex) {  ex.printStackTrace();  }  }  });  addWindowListener(new WindowAdapter() {  @Override  public void windowClosing(WindowEvent e) {  try {  client.close();  } catch (IOException ex) {  ex.printStackTrace();  }  System.exit(0);  }  });  setVisible(true);  }  void init() {  setLocation(500,10);  setSize(400,400);  setResizable(false);  setBackground(Color.MAGENTA);  TA.setBackground(Color.white);  Panel Up = new Panel();  Up.add(new Label("IP"));  Up.add(this.ip);  Up.add(new Label("Port"));  Up.add(this.port);  Up.add(this.connect);  this.add(Up, BorderLayout.NORTH);  Panel Mid = new Panel();  Mid.add(this.TA);  this.add(Mid, BorderLayout.CENTER);  Panel Down = new Panel();  Down.add(new Label("Send"));  Down.add(this.TF);  Down.add(this.send);  this.add(Down, BorderLayout.SOUTH);  pack();  }  class ClientThread extends Thread{  @Override  public void run() {  try {  BufferedReader br = new BufferedReader(new InputStreamReader(client.getInputStream()));  String s = br.readLine();  if (s == "Server send:bye"){  client.close();  }  while (!"Server send:bye".equals(s) && s != null){  TA.append(s + "\n");  s = br.readLine();  }  } catch (IOException e) {  e.printStackTrace();  }  }  }  }  public class SocketClient {  public static void main(String[] args) {  myClientFrame cf = new myClientFrame("客户端");  }  }  ServerSocket.java  package pratice3;  import java.awt.\*;  import java.awt.event.ActionEvent;  import java.awt.event.ActionListener;  import java.awt.event.WindowAdapter;  import java.awt.event.WindowEvent;  import java.io.BufferedReader;  import java.io.IOException;  import java.io.InputStreamReader;  import java.io.PrintWriter;  import java.net.ServerSocket;  import java.net.Socket;  //窗体  class myServerFrame extends Frame{  ServerSocket server = null;  Socket client = null;  TextField port = new TextField("8888");  Button start = new Button("Start");  TextArea TA = new TextArea(25, 60);  TextField TF = new TextField(40);  Button send = new Button("Send");  myServerFrame(String title) {  super(title);  init(); //初始化窗口  start.addActionListener(new ActionListener() {  @Override  public void actionPerformed(ActionEvent e) {  try {  start.setEnabled(false);  int i = Integer.parseInt(port.getText());  TA.append("Server Port:" + port.getText() + "\n");  server = new ServerSocket(i);  client = server.accept();  TA.append("A client connect...\n");  ServerThread ST = new ServerThread();  ST.start();  } catch (IOException ex) {  ex.printStackTrace();  }  }  });  send.addActionListener(new ActionListener() {  @Override  public void actionPerformed(ActionEvent e) {  try {  PrintWriter s = new PrintWriter(client.getOutputStream());  String ss = TF.getText();  if (ss != ""){  s.write("Server send:" + ss + "\n");  TA.append("Server send:" + ss + "\n");  s.flush();  }  TF.setText("");  } catch (IOException ex) {  ex.printStackTrace();  }  }  });  this.addWindowListener(new WindowAdapter() {  @Override  public void windowClosing(WindowEvent e) {  try {  server.close();  } catch (IOException ex) {  ex.printStackTrace();  }  System.exit(0);  }  });  this.setVisible(true);  }  void init() {  setLocation(10,10);  setSize(400,400);  setResizable(false);  setBackground(Color.MAGENTA);  TA.setBackground(Color.white);  Panel Up = new Panel();  Up.add(new Label("Port"));  Up.add(this.port);  Up.add(this.start);  this.add(Up, BorderLayout.NORTH);  Panel Mid = new Panel();  Mid.add(this.TA);  this.add(Mid, BorderLayout.CENTER);  Panel Down = new Panel();  Down.add(new Label("Send"));  Down.add(this.TF);  Down.add(this.send);  this.add(Down, BorderLayout.SOUTH);  pack();  }  class ServerThread extends Thread{  @Override  public void run() {  try {  BufferedReader br = new BufferedReader(new InputStreamReader(client.getInputStream()));  String s = br.readLine();  if (s == "Client send:bye"){  client.close();  }  while (!"Client send:bye".equals(s) && s != null){  TA.append(s + "\n");  s = br.readLine();  }  } catch (IOException e) {  e.printStackTrace();  }  }  }  }  public class SocketServer {  public static void main(String[] args) {  myServerFrame frame = new myServerFrame("服务端");  }  }    分析：  在服务器端的类中，通过使用ServerSocket类的accept方法返回一个socket类的对象，通过这个对象与客户端类利用IP和端口号进行连接，在客户端通过对应的IP和端口号与服务器连接。连接成功之后利用Thread类同时进行输入输出的操作，通过流类进行客户端和服务器之间的交流。  **五、实验结论、分析、思考题与心得体会**  通过本次实验，我有以下几方面的收获、认识和体会：  1.加深了对于Java中系统类的理解，例如静态函数，返回值的类型等等  2.对于GUI界面的编写更加的熟悉，学会了利用不同的布局管理器、Panel、事件监听器等等  3.学会了利用Socket和ServerSocket类进行客户端和服务器之间的简单通信 | | | | | | | | |