**Java程序设计实验报告**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 姓名 | 吕泽 | 学号 | 55151031 | 成绩 |  |
| 实验1 | | | | | |
| 实验内容 请编写一个Java程序输出自己的姓名、年龄、身高和体重，分别通过控制台和Eclipse环境运行 | | | | | |
| 实验源码和注释  public class 作业1 {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  System.out.print("姓名：吕泽\n");  System.out.print("年龄：19\n");  System.out.print("身高：185cm\n");  System.out.print("体重：85kg\n");  }    } | | | | | |
| 关键问题解析  使用类内静态main函数输出，调用System的方法。 | | | | | |
| 实验2 | | | | | |
| 实验内容 打印出100到1000之间所有的“水仙花数”，所谓“水仙花数”是指一个三位数，其各位数字立方和等于该数本身。例如：153是一个“水仙花数”，因为153=1³+5³+3³。 | | | | | |
| 实验源码和注释  public class JavaApplication3 {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  int a,b,c;  for(int i=100;i<1000;i++)  {  a=i/100;  b=(i-a\*100)/10;  c=(i-a\*100-b\*10);  if(a\*a\*a+b\*b\*b+c\*c\*c==i)  System.out.println(""+i+"是水仙花数");  }  }    } | | | | | |
| 关键问题解析  使用int类型赋值时小数舍去的特性来获得三位数的各位数字的值。 | | | | | |
| 实验3 | | | | | |
| 实验内容  利用Java类库中的Point类设计一个三角形类（Triangle），编程求三角形面积。要求写出源程序、执行步骤和结果。 | | | | | |
| 实验源码和注释  package javaapplication4;  import java.lang.Math;  import java.awt.\*;  /\*\*  \*  \* @author VE  \*/  public class JavaApplication4 {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  Triangle llll=new Triangle(new Point(0,0),new Point(3,0),new Point(0,4));  System.out.print(llll.area());  }  }  class Triangle{  Point a;  Point b;  Point c;  Triangle (Point aa,Point bb,Point cc)  {  a=aa;b=bb;c=cc;  }  public float area()  {  float aa;  aa = (float) Math.sqrt(Math.pow(Math.abs(a.x-b.x),2)+Math.pow(Math.abs(a.y-b.y),2));//ab之间  float bb;  bb = (float) Math.sqrt(Math.pow(Math.abs(c.x-b.x),2)+Math.pow(Math.abs(c.y-b.y),2));//bc  float cc;  cc = (float) Math.sqrt(Math.pow(Math.abs(a.x-c.x),2)+Math.pow(Math.abs(a.y-c.y),2));//ac  float p=(aa+bb+cc)/2;  return (float) Math.sqrt(p\*(p-aa)\*(p-bb)\*(p-cc));  }  } | | | | | |
| 关键问题解析  使用Point类作为三角形类的成员变量，以标注各点位置。计算面积时用到了海伦——秦九昭公式，然后将计算结果返回。  Main函数中测试了一个以（0，0），（3，0），（0，4）作为三顶点的三角形。 | | | | | |
| 实验4 | | | | | |
| 实验内容  为某研究所编写一个通用程序，用来计算每一种交通工具运行1000公里所需的时间，已知每种交通工具的参数都是3个整数A、B、C的表达式。现有两种工具：Car007 和Plane，其中Car007 的速度运算公式为：A\*B/C，Plane 的速度运算公式为：A+B+C。需要编写三类：ComputeTime.java,Plane.java,Car007.java和接口Common.java，要求在未来如果增加第3种交通工具的时候，不必修改以前的任何程序，只需要编写新的交通工具的程序。其运行过程如下，从命令行输入ComputeTime的四个参数，第一个是交通工具的类型，第二、三、四个参数分别时整数A、B、C，举例如下：计算Plane的时间："java ComputeTime Plane 20 30 40"计算Car007的时间："java ComputeTime Car007 23 34 45"如果第3种交通工具为Ship,则只需要编写Ship.java，运行时输入："javaComputeTime Ship 22 33 44"   提示：充分利用接口的概念，接口对象充当参数。实例化一个对象的另外一种办法：Class.forName(str).newInstance（）；例如需要实例化一个Plane对象的话，则只要调用Class.forName("Plane").newInstance()便可。 | | | | | |
| 实验源码和注释  Car007.java：  package javaapplication5;  public class Car007 implements Common{  Car007(){};  public int v(int A,int B,int C)  {  return A\*B/C;  }  }  Plane.java：  package javaapplication5;  /\*\*  \*  \* @author VE  \*/  public class Plane implements Common{  Plane(){};  public int v(int A,int B,int C)  {  return A+B+C;  }  }  Common.java:  package javaapplication5;  public interface Common {    public int v(int A,int B,int C);  }  Computertime.java:  package javaapplication5;  import java.util.Scanner;  public class ComputerTime {    public static void main(String[] args) {  Scanner scanner= new Scanner(System.in);  String s;    s = scanner.next();  s = "javaapplication5."+s;  int A,B,C;  A = scanner.nextInt();  B = scanner.nextInt();  C = scanner.nextInt();  scanner.close();  Common transport = null;  try {  transport = (Common) Class.forName(s).newInstance();  } catch (InstantiationException e) {    e.printStackTrace();  } catch (IllegalAccessException e) {    e.printStackTrace();  } catch (ClassNotFoundException e) {    e.printStackTrace();  }  System.out.println(transport.v(A, B, C));  }  } | | | | | |
| 关键问题解析  四个文件中Common是定义借口的文件，ComputerTime是最终实现的文件，Car007和Plane中定义了各自的接口函数。  在CompuerTime中使用了Class.forName（str）.newInstance（）方法实例化新对象，其中类型使用的用户输入的类型名。但是需要注意的是类型str需要加上包名才能实现调用。然后使用一个接口类型变量接收该实例对象，调用其方法计算结果。 | | | | | |
| 实验5 | | | | | |
| 实验内容  .编写程序ColorPane.java，实现下面的界面布局效果，并对每个按钮加载监听器，使得当按下一个按钮时，这个按钮的表面显现出与其上面所写的名字相同的颜色。  http://cc.jlu.edu.cn/G2S/eWebEditor/uploadfile/20131120161125001.jpg      http://cc.jlu.edu.cn/G2S/eWebEditor/uploadfile/20131120161125002.jpg | | | | | |
| 实验源码和注释  package 按钮变色;  frame.java:  import java.awt.Color;  public class frame extends javax.swing.JFrame {  /\*\*  \* Creates new form 窗体  \*/  public frame() {  initComponents();  }  /\*\*  \* This method is called from within the constructor to initialize the form.  \* WARNING: Do NOT modify this code. The content of this method is always  \* regenerated by the Form Editor.  \*/  @SuppressWarnings("unchecked")  // <editor-fold defaultstate="collapsed" desc="Generated Code">  private void initComponents() {  blue = new javax.swing.JButton();  cyan = new javax.swing.JButton();  green = new javax.swing.JButton();  pink = new javax.swing.JButton();  yellow = new javax.swing.JButton();  red = new javax.swing.JButton();  white = new javax.swing.JButton();  magenta = new javax.swing.JButton();  orange = new javax.swing.JButton();  setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE);  blue.setText("blue");  blue.addActionListener(new java.awt.event.ActionListener() {  public void actionPerformed(java.awt.event.ActionEvent evt) {  blueActionPerformed(evt);  }  });  cyan.setText("cyan");  cyan.addActionListener(new java.awt.event.ActionListener() {  public void actionPerformed(java.awt.event.ActionEvent evt) {  cyanActionPerformed(evt);  }  });  green.setText("green");  green.addActionListener(new java.awt.event.ActionListener() {  public void actionPerformed(java.awt.event.ActionEvent evt) {  greenActionPerformed(evt);  }  });  pink.setText("pink");  pink.addActionListener(new java.awt.event.ActionListener() {  public void actionPerformed(java.awt.event.ActionEvent evt) {  pinkActionPerformed(evt);  }  });  yellow.setText("yellow");  yellow.addActionListener(new java.awt.event.ActionListener() {  public void actionPerformed(java.awt.event.ActionEvent evt) {  yellowActionPerformed(evt);  }  });  red.setText("red");  red.addActionListener(new java.awt.event.ActionListener() {  public void actionPerformed(java.awt.event.ActionEvent evt) {  redActionPerformed(evt);  }  });  white.setText("white");  white.addActionListener(new java.awt.event.ActionListener() {  public void actionPerformed(java.awt.event.ActionEvent evt) {  whiteActionPerformed(evt);  }  });  magenta.setText("magenta");  magenta.addActionListener(new java.awt.event.ActionListener() {  public void actionPerformed(java.awt.event.ActionEvent evt) {  magentaActionPerformed(evt);  }  });  orange.setText("orange");  orange.addActionListener(new java.awt.event.ActionListener() {  public void actionPerformed(java.awt.event.ActionEvent evt) {  orangeActionPerformed(evt);  }  });  javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());  getContentPane().setLayout(layout);  layout.setHorizontalGroup(  layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  .addGroup(layout.createSequentialGroup()  .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING, false)  .addComponent(blue, javax.swing.GroupLayout.Alignment.LEADING, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)  .addComponent(magenta, javax.swing.GroupLayout.Alignment.LEADING, javax.swing.GroupLayout.DEFAULT\_SIZE, 132, Short.MAX\_VALUE)  .addComponent(red, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))  .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)  .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)  .addComponent(cyan, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)  .addComponent(orange, javax.swing.GroupLayout.DEFAULT\_SIZE, 132, Short.MAX\_VALUE)  .addComponent(white, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))  .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)  .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)  .addComponent(green, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)  .addComponent(pink, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)  .addComponent(yellow, javax.swing.GroupLayout.DEFAULT\_SIZE, 132, Short.MAX\_VALUE))  .addContainerGap())  );  layout.setVerticalGroup(  layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  .addGroup(layout.createSequentialGroup()  .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  .addComponent(blue, javax.swing.GroupLayout.PREFERRED\_SIZE, 99, javax.swing.GroupLayout.PREFERRED\_SIZE)  .addComponent(cyan, javax.swing.GroupLayout.PREFERRED\_SIZE, 99, javax.swing.GroupLayout.PREFERRED\_SIZE)  .addComponent(green, javax.swing.GroupLayout.PREFERRED\_SIZE, 99, javax.swing.GroupLayout.PREFERRED\_SIZE))  .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)  .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)  .addComponent(magenta, javax.swing.GroupLayout.DEFAULT\_SIZE, 99, Short.MAX\_VALUE)  .addComponent(orange, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)  .addComponent(pink, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))  .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)  .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)  .addComponent(red, javax.swing.GroupLayout.DEFAULT\_SIZE, 99, Short.MAX\_VALUE)  .addComponent(white, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)  .addComponent(yellow, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))  .addContainerGap(javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))  );  pack();  }// </editor-fold>  private void blueActionPerformed(java.awt.event.ActionEvent evt) {  blue.setBackground(Color.blue);  }  private void cyanActionPerformed(java.awt.event.ActionEvent evt) {  cyan.setBackground(Color.cyan);  }  private void greenActionPerformed(java.awt.event.ActionEvent evt) {  green.setBackground(Color.green);  }  private void magentaActionPerformed(java.awt.event.ActionEvent evt) {  magenta.setBackground(Color.magenta);  }  private void orangeActionPerformed(java.awt.event.ActionEvent evt) {  orange.setBackground(Color.orange);  }  private void pinkActionPerformed(java.awt.event.ActionEvent evt) {  pink.setBackground(Color.pink);  }  private void redActionPerformed(java.awt.event.ActionEvent evt) {  red.setBackground(Color.red);  }  private void whiteActionPerformed(java.awt.event.ActionEvent evt) {  white.setBackground(Color.white);  }  private void yellowActionPerformed(java.awt.event.ActionEvent evt) {  yellow.setBackground(Color.yellow);  }  /\*\*  \* @param args the command line arguments  \*/  public static void main(String args[]) {  /\* Set the Nimbus look and feel \*/  //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">  /\* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.  \* For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html  \*/  try {  for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {  if ("Nimbus".equals(info.getName())) {  javax.swing.UIManager.setLookAndFeel(info.getClassName());  break;  }  }  } catch (ClassNotFoundException ex) {  java.util.logging.Logger.getLogger(frame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);  } catch (InstantiationException ex) {  java.util.logging.Logger.getLogger(frame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);  } catch (IllegalAccessException ex) {  java.util.logging.Logger.getLogger(frame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);  } catch (javax.swing.UnsupportedLookAndFeelException ex) {  java.util.logging.Logger.getLogger(frame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);  }  //</editor-fold>  //</editor-fold>  /\* Create and display the form \*/  java.awt.EventQueue.invokeLater(new Runnable() {  public void run() {  new frame().setVisible(true);  }  });  }  // Variables declaration - do not modify  private javax.swing.JButton blue;  private javax.swing.JButton cyan;  private javax.swing.JButton green;  private javax.swing.JButton magenta;  private javax.swing.JButton orange;  private javax.swing.JButton pink;  private javax.swing.JButton red;  private javax.swing.JButton white;  private javax.swing.JButton yellow;  // End of variables declaration  }  按钮变色.java:  package 按钮变色;  import javax.swing.JFrame;  public class 按钮变色 {    public static void main(String[] args) {  frame a= new frame();  a.setVisible(true);  a.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);    }    } | | | | | |
| 关键问题解析  创建按钮、编辑样式。为每一个按钮添加监听器以实行动作，使用Color.cyan（灰色为例）获得对应颜色的属性值作为参数放入按钮类的setBackground()方法中改变颜色。 | | | | | |
| 实验6 | | | | | |
| 实验内容  输入若干整数，并求和，直到结束标志999为止。在输入过程中，自动忽略掉输入的非整数。  修改以上程序，通过对负数输入产生异常的方式，对输入的负数不进行累加运算。 | | | | | |
| 实验源码和注释  整数求和.java:  package 整数求和;  import java.util.Scanner;  public class 整数求和 {  public static void main(String[] args) {  int sum=0;  Scanner scanner;  int add=0;    scanner = new Scanner(System.in);  while(true){  try{  add=scanner.nextInt();  if(add==999)  break;  if(add<0)  throw new nagetivenumber();  sum+=add;  }catch(java.util.InputMismatchException re){  scanner.next();  }  }  System.out.print("值为"+sum+"");    }  }  nagetivenumber.java:  package 整数求和;  public class nagetivenumber extends java.util.InputMismatchException{  public nagetivenumber(){    }} | | | | | |
| 关键问题解析  新建了一个nagetivenumber的异常，当识别到输入的数为负数时会引发该异常，而scanner.nextInt()本身当输入数值非int时也会引发内建异常。新建异常是该内建异常的子类，检测时只检测内建异常类即可全部检查。  当触发异常捕获后需要将改引发异常的非正整数值跳过以避免下一次nextInt会再次读该数据继续循环异常，此处使用scanner.next（）方法跳过。 | | | | | |
| 实验7 | | | | | |
| 实验内容  使用 List 的任一实现类构造一个队列数据结构，需使用泛型。  要求：必须实现下列方法：  get()：出队列，返回出队列的结点  put()：入队列  isEmpty()：判断队列是否为空 | | | | | |
| 实验源码和注释  Queue.java  package 实验三;  import java.util.LinkedList;  /\*\*  \*  \* @author VE  \* @param <T> 类型参数  \*/  public class queue <T>{  public queue(){  x=new LinkedList();  }  LinkedList<T> x;  public T get(){  return x.poll();  }  public void add(T a){  x.add(a);  }  public boolean isEmpty()  {  return x.isEmpty();  }  }  实验三.java  package 实验三;  public class 实验三 {  public static void main(String[] args) {  queue<Integer> x=new queue();  x.add(1);  x.add(2);  x.add(3);  System.out.print(x.get());  System.out.print(x.get());  x.add(5);  System.out.print(x.isEmpty());    }    } | | | | | |
| 关键问题解析  使用了双向链表类LinkedList作为成员变量构建queue类，通过双向链表的poll、add和isEmpty方法作为queue的出、入队列和是否为空方法的实现方式。  在Main中实例化了该类并进行了一些测试。 | | | | | |
| 实验8 | | | | | |
| 实验内容  设计一个火车票模拟程序Ticket.java。假如火车站有100张火车票要卖出，现在有5个售票点同时售票，用5个线程模拟这5个售票点的售票情况。要求如下：  1）打印出每个售票点所卖出的票号；  2）各售票点不能售出相同票号的火车票。 | | | | | |
| 实验源码和注释  Ticketbase.java:  package 实验三第二题;  import java.util.logging.Level;  import java.util.logging.Logger;  public class Ticketbase implements Runnable{  public Ticketbase(int x){rest=x;}  int rest;  public void run()  {  while(rest!=0)  {  synchronized(this)  { System.out.println(Thread.currentThread().getName()+ "\_\_\_\_\_\_\_\_"+rest--+"\n");}  try {  Thread.sleep((int)Math.random()\*10);  } catch (InterruptedException ex) {  Logger.getLogger(Ticketbase.class.getName()).log(Level.SEVERE, null, ex);  }    }  }  }  实验三第二题.java  package 实验三第二题;  public class 实验三第二题 {  public static void main(String[] args) {  Ticketbase x=new Ticketbase(100);  new Thread(x,"1").start();  new Thread(x,"2").start();  new Thread(x,"3").start();  new Thread(x,"4").start();  new Thread(x,"5").start();  }  } | | | | | |
| 关键问题解析  构建了一个实现Runable接口的Ticketbase类，将Run（）方法实现。其中使用了synchronized(this){}锁的方式避免多个线程同时进入核心部分以避免打印顺序错乱。Main函数中新建了五个同一实例化的该类运行。 | | | | | |
| 实验9 | | | | | |
| 实验内容  设计基于流套接字的客户机/服务器通信程序。要求如下：   1. 分别编写服务端程序和客户端程序（端口号用1888）. 2. 客户端向服务器发送信息，雾浮起接受到信息并显示出来。 3. 同时服务器将信息反馈到客户端，客户端收到此此信息并显示。 | | | | | |
| 实验源码和注释  客户端.java  package 通信;  import java.io.BufferedReader;  import java.io.DataOutputStream;  import java.io.IOException;  import java.io.InputStreamReader;  import java.net.InetAddress;  import java.net.Socket;  import java.util.Scanner;  /\*\*  \*  \* @author VE  \*/  public class 客户端 {  Socket socket;  public 客户端() throws IOException {  this.socket = new Socket(InetAddress.getLoopbackAddress(),1888);  }  public void go() throws IOException, InterruptedException{  // boolean autoflush=true;  //PrintWriter out =new PrintWriter(socket.getOutputStream(),autoflush);  System.out.print("这里是客户端！\n");  DataOutputStream output=new DataOutputStream(socket.getOutputStream());  BufferedReader input=new BufferedReader(new InputStreamReader(socket.getInputStream()));//读取控制命令  //发送消息  Scanner scanner=new Scanner(System.in);  System.out.print("请输入您要传递的信息：");  String x=scanner.next();  output.writeUTF(x);  //下面是接受回应  String cmd=input.readLine();  System.out.print("回应信息："+cmd);  }  static public void main(String[] args) throws IOException, InterruptedException  {  new 客户端().go();  }  }  服务端.java  package 通信;  import java.io.DataInputStream;  import java.io.IOException;  import java.io.OutputStreamWriter;  import java.io.Writer;  import java.net.ServerSocket;  import java.net.Socket;  import java.util.Scanner;  import java.util.logging.Level;  import java.util.logging.Logger;  public class 服务端 {  ServerSocket serversocket;    public 服务端() throws IOException {  this.serversocket = new ServerSocket(1888);  }  public void start() throws IOException  {  Socket socket=serversocket.accept();  new ServerThead(socket).start();  }    public class ServerThead extends Thread{  Socket socket;  ServerThead(Socket socket){  this.socket=socket;  }  public void run(){  System.out.print("这里是服务端！\n");  try {  Writer output=new OutputStreamWriter(socket.getOutputStream());//------  DataInputStream input=new DataInputStream(socket.getInputStream());  System.out.println("连接成功！客户信息："+input.readUTF());  Scanner scanner=new Scanner(System.in);  System.out.print("请输入您要回应的信息：");  output.write(scanner.next());  output.flush();  socket.close();  } catch (IOException ex) {  Logger.getLogger(服务端.class.getName()).log(Level.SEVERE, null, ex);  }  try {  serversocket.close();  } catch (IOException ex) {  Logger.getLogger(服务端.class.getName()).log(Level.SEVERE, null, ex);  }  }  }  static public void main(String[] args) throws IOException{  new 服务端().start();  }  } | | | | | |
| 关键问题解析  构建了两个java类，一个是使用serversocket类实现的服务端，一个是使用socket类实现的客户端。客户端中socket变量绑定1888端口并调用了getOutputStream()方法和getInputStream()方法获取输出和输入数据的流。在服务端中有serversocker绑定1888端口后调用accept（）方法获取了连接到1888端口的socket类并定义了一份socket类。然后使用该socket的getOutputStream()和getInputStream()方法获取输出和输入数据的流。  然后在客户端以及服务端分别使用了BufferedReader，DataOutputStream和DataInputStream，Writer类实例化后进行传输、获取数据。 | | | | | |
| 总体感想  Java语言拥有强大的包和功能，熟练掌握后为程序的编写和运行创造了非常完美的运行条件。 | | | | | |