**《Java技术》实验报告**

实验三： 2020 年04 月10 日

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 学院 | 计算机与信息学院 | 专业班级 | 物联网工程19-2班 | 姓名 | 袁焕发 | 成绩 |  |
| 课程  名称 | Java技术 | 实验项目  名 称 | 基于GUI的网络通信程序设计 | | | 指导 教师 | 薛峰 |
| 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  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。  **思考题**  1. 对整个实验进行总结，写出实验心得。  2. 在完成上述实验内容的基础上，尝试实现“双向通信”功能，即服务端、客户端之间可以相互发送、接收消息，并以此作为实验成绩评优的加分依据。 | | | | | | |
|  | 三、使用硬件、软件环境  PC 计算机一台，配置为CPU为2.4G，内存为16G，硬盘为500G，安装Windows10操作系统。  另外，使用Notepad++，Eclipse ,JDK1.8.0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)  源代码  package testwork3;  import java.awt.\*;  import java.awt.event.\*;  import java.io.\*;  import java.net.\*;  import javax.swing.\*;  public class Server{  public static void main(String[] args){  MyFrame server = new MyFrame();  server.setTitle("服务器");  server.setVisible(true); //窗口可见  server.setResizable(false); //不可调整  server.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); //设置默认关闭操作  }  }  class MyFrame extends JFrame{  JTextField port;  JButton start;  JTextArea content;  JTextField cin;  JButton say;  Socket socket;  MyFrame(){  init();  StartListen sListen = new StartListen();  SayListen stListen = new SayListen();  start.addActionListener(sListen); //为按钮 start 注册一个监听器  say.addActionListener(stListen); //为按钮 say 注册一个监听器  }    void init(){  //设置面板位置及其属性  JPanel jpanel=new JPanel();  add(jpanel,BorderLayout.NORTH);  setSize(500,500);  setLocation(400,100);  //设置四个行型容器一个竖型容器  Box box1=Box.createHorizontalBox();  Box box2=Box.createHorizontalBox();  Box box3=Box.createHorizontalBox();  Box box4=Box.createHorizontalBox();  Box box5=Box.createVerticalBox();  //设置两个标签，一个文本框一个按钮  JLabel s1=new JLabel("端口设置Port:");  JLabel s2=new JLabel("启动服务器");  port = new JTextField("8888",8);  start = new JButton("Start");  box1.add(s1);  box1.add(port);  box1.add(s2);  box1.add(start);  JLabel s3=new JLabel("聊天框");  box2.add(s3);  content = new JTextArea(18,40);  content.setBackground(Color.lightGray);//设置背景颜色  JScrollPane scroll = new JScrollPane(content); //设置滚动条  box3.add(scroll);  JLabel s4=new JLabel("信息输入框 ");  box4.add(s4);  cin = new JTextField(26);  box4.add(cin);  say = new JButton("发送");  box4.add(say);  box5.add(box1);  box5.add(box2);  box5.add(box3);  box5.add(box4);  //容器添加到面板  jpanel.add(box5);  }    //监听服务器启动  class StartListen implements ActionListener{  public void actionPerformed(ActionEvent e){  start.setEnabled(false);  try {  ServerSocket s = new ServerSocket(Integer.parseInt(port.getText()));  // 创建一个服务器对象 s读取端口号port  socket = s.accept(); //接收  PrintWriter out = new PrintWriter(new BufferedWriter(new OutputStreamWriter(socket.getOutputStream())),true);  // 创建一个打印输出流，从 socket对象获取输出流  out.println("Connected"); //向客户端输出信息  content.append("Client connected"+"\n"); //如果连接成功则服务器界面显示  ServerThread st = new ServerThread();  //创建一个 ServerThread 对象来获取socket信息  st.start(); //启动线程  s.close;    }  catch (Exception ex) {  }  }  }  //监听信息发送  class SayListen implements ActionListener{  String str;  public void actionPerformed(ActionEvent e){  try {  PrintWriter out = new PrintWriter(new BufferedWriter(new OutputStreamWriter(socket.getOutputStream())),true);  // 创建一个打印输出流，从socket 获取输出流  str=cin.getText(); //获取字符串  if(!str.isEmpty()){  out.println("对方："+str); //向客户端输出  content.append( "我："+str+"\n");//服务器界面显示  out.flush(); //清空缓存区  }  }  catch (Exception ex) {  }  }  }  //接收来自客户端的信息流  class ServerThread extends Thread{  public void run(){  try {  BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));  //创建一个缓冲输出流，其形参为从套接字 socket 中获取的输入流  String str;  while(true){  str = in.readLine(); //按行读取  content.append(str+"\n"); //显示对方的信息  }  }  catch (Exception ex) {  }  }  }  }  package testwork3;  import java.awt.\*;  import java.awt.event.\*;  import java.io.\*;  import java.net.\*;  import javax.swing.\*;  public class Client{  public static void main(String[] args)  {  ClientFrame client=new ClientFrame();  client.setTitle("客户端");  client.setVisible(true);  client.setResizable(false); //设置此窗体是否可由用户调整大小  client.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); // 设置关闭操作  }    }  class ClientFrame extends JFrame {    JTextField ip;  JTextField port;  JTextField cin;  JTextArea content;  JButton connect;  JButton say;  Socket socket;  ClientFrame(){  init();  ConnectListen cListener=new ConnectListen();  SayListen sListener = new SayListen();  connect.addActionListener(cListener); //  say.addActionListener(sListener); //注  }  void init(){  JPanel jPanel=new JPanel();  add(jPanel,BorderLayout.NORTH);  setSize(500,500);  setLocation(900,100);  Box box1=Box.createHorizontalBox();  Box box2=Box.createHorizontalBox();  Box box3=Box.createHorizontalBox();  Box box4=Box.createHorizontalBox();  Box box5=Box.createVerticalBox();  JLabel jLabel1=new JLabel("服务器IP");  ip=new JTextField("127.0.0.1",8);  box1.add(jLabel1);  box1.add(ip);  JLabel jLabel2=new JLabel("服务器端口");  port=new JTextField("8888",8);  connect=new JButton("connect");  box1.add(jLabel2);  box1.add(port);  box1.add(connect);  JLabel jLabel3=new JLabel("聊天框");  box2.add(jLabel3);  content=new JTextArea(18,40);  content.setBackground(Color.cyan);  JScrollPane scroll=new JScrollPane(content);  box3.add(scroll);  JLabel jLabel4=new JLabel("信息发送框");  box4.add(jLabel4);  cin=new JTextField(20);  box4.add(cin);  say=new JButton("发送");  box4.add(say);  box5.add(box1);  box5.add(box2);  box5.add(box3);  box5.add(box4);  //容器添加到面板  jPanel.add(box5);  }  //连接监听器  class ConnectListen implements ActionListener{  int portNum;  public void actionPerformed(ActionEvent e){  connect.setEnabled(false); //按下 connect按钮后不可再次点击  try {  content.setText("Connect to server....\n"); //客户端界面显示  portNum =Integer.parseInt(port.getText()); //读取 端口字符串，并将字符串转化为整型  socket = new Socket(ip.getText(),portNum); //定义对象 socket，并指定IP和端口  ClientThread c= new ClientThread(); //定义一个对象  c.start(); //启动线程  }  catch (Exception ex) {  }  }  }  //发送信息监听器  class SayListen implements ActionListener{  String str;  public void actionPerformed(ActionEvent e){  try {  PrintWriter out = new PrintWriter(new BufferedWriter(new OutputStreamWriter(socket.getOutputStream())),true);  //从 socket 获取字节输出流，并将其最终转化为 PrintWriter  str=cin.getText(); //从文本框中获取字符串  if(!str.isEmpty()){  out.println("对方："+str); //向服务器发送信息  content.append("我： "+str+"\n"); //客户端显示自己发送的信息  out.flush(); //清空缓存区  }  }  catch (Exception ex) {  }  }  }  //监听来自服务器的信息  class ClientThread extends Thread{  public void run(){  try {  BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));  // 获取 socket输入流，并将字节流最终转换为缓冲字符输入流  String str;  while(true){  str = in.readLine(); //按行读取  content.append(str+"\n"); //发送来的信息  }  }  catch (Exception ex) {  }  }  }  }      五、实验结论、分析、思考题与心得体会  这次的实验很难，因为涉及到了GUI的编写，线程，Socket类的使用，输入输出流的使用。首先在界面创建上，一开始是直接在一个面板上添加所有的标签，按钮，文本框，但是一直无法使得界面排列有序美观，后来查阅资料创建了几个行型box，一个列型box,先在box中有序排列好再添加到面板。之后是ServerSocket对象以及客户端套接字的构建，这花了我大部分时间，首先不断根据课本的例题进行理解，又把课程的视频录像进行反复回放，  之后才进行了编写，然后查找代码错误修改代码花了很久，GUI和通信这两方面是分开的，所以我又要想办法创建监视器监视按钮的操作情况，将有操作的情况与通信的建立，信息的发送联系起来，将两者匹配又要修改很多的代码。这次的实验做的很累，但是对于GUI的编写有了进一步的认识，同时对于Socket类和ServerSocket的应用能力需要提高。 | | | | | | |