**《Java程序开发》实验报告**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **姓名** | | **任齐轩** | | | **年级** | **2020级** |
| **学号** | | **20204154** | | | **专业、班级、** | **计卓2班** |
| **实验名称** | **用socket实现双人聊天程序** | | | | | |
| **实验时间** | **2021.11.11** | | **实验地点** | **DS1408** | | |
| **实验成绩** |  | | **实验性质** | **□验证性 □设计性 □综合性** | | |
| 教师评价：  评价教师签名： | | | | | | |
| 一、实验目的  用TCP的socket编程实现简单的双人聊天程序，要分别实现服务端和客户端。要实现两个方面的功能：1能进行文字对话；2能传输任意文件文件。  要求：有GUI界面，在界面上输入文字，能看对方发送的文字；在界面上选择文件发送，对方接收后保存到自己事先指定的目录中。为了简化，目录可以在程序中固定。传输文件时可考虑约定另外的固定端口号，即文字聊天和文件传输可用不同的端口来进行。 | | | | | | |
| 二、实验项目内容  用TCP的socket编程实现简单的双人聊天程序，要分别实现服务端和客户端。要实现两个方面的功能：1能进行文字对话；2能传输任意文件文件。  要求：有GUI界面，在界面上输入文字，能看对方发送的文字；在界面上选择文件发送，对方接收后保存到自己事先指定的目录中。为了简化，目录可以在程序中固定。传输文件时可考虑约定另外的固定端口号，即文字聊天和文件传输可用不同的端口来进行。 | | | | | | |
| 1. 实验过程或算法（源程序）   服务端：  import javax.swing.\*; import java.awt.\*; import java.awt.event.ActionEvent; import java.awt.event.ActionListener; import java.io.\*; import java.net.ServerSocket; import java.net.Socket;   //1.define every blocks in frame //2.initialize all blocks public class ServerChatMain extends JFrame implements ActionListener {    public static void main(String[] args) {  new ServerChatMain();//initialize  }   //text area  private JTextArea jta;  //scroll pane  private JScrollPane jsp;  //blank space  private JPanel jp;  //text box  private JTextField jtf;   //button  private JButton jb;  //file send button  private JButton jsend;   // output stream  private BufferedWriter bw = null;   //creation  public ServerChatMain(){  //initial all block  jta = new JTextArea();  //set the text uneditable  jta.setEditable(false);  //add the text to the scroll pane to scroll  jsp = new JScrollPane(jta);  //blank space  jp = new JPanel();  //text box  jtf = new JTextField(10);  //button  jb = new JButton("Send");  //file sending button  jsend = new JButton("Send File");   //Add buttons to blank space  jp.add(jtf);  jp.add(jb);  jp.add(jsend);    //add everything into window  this.add(jsp, BorderLayout.*CENTER*); //put in the middle  this.add(jp,BorderLayout.*SOUTH*); //put in the bottom  this.add(jsend,BorderLayout.*NORTH*);   // set everything visible  this.setTitle("CTalk-Server");  this.setSize(300,300);  this.setLocation(300,300);   this.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  this.setVisible(true);   jb.addActionListener(this);  try{  ServerSocket serversocket = new ServerSocket(7777);    Socket socket = serversocket.accept();   BufferedReader br = new BufferedReader(new InputStreamReader(socket.getInputStream()));  String line = null;   bw = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()));     while((line = br.readLine()) != null) {   if (line.contains("[File\_request]")) {  jta.append("A File will be sent to you" + System.*lineSeparator*());  jta.append("File Name: " + line.split("#")[1] + System.*lineSeparator*());  int port = 9999;  FileThread filethread = new FileThread(port);  filethread.start();  System.*out*.println("File Thread start");  jta.append("File Transfer succeeded! Saved in: \"/Users/calvin/ \" " + System.*lineSeparator*());  bw.append("File Transfer succeeded!");  bw.newLine();  bw.flush();  }   else jta.append(line + System.*lineSeparator*());  }  serversocket.close();   } catch (Exception e2) {  e2.printStackTrace();  }   // extends a interface    jsend.addActionListener(new ActionListener() {  @Override  public void actionPerformed(ActionEvent e) {  JFrame frame = new JFrame();   JFileChooser sourceFileChooser = new JFileChooser(".");  sourceFileChooser.setFileSelectionMode(JFileChooser.*FILES\_ONLY*);  int status = sourceFileChooser.showOpenDialog(frame);  File sourceFile = new File(sourceFileChooser.getSelectedFile().getPath());   jta.append("Send the file: <" + sourceFile.getName() + ">\r\n");  Socket socket2 = null;  try {  socket2 = new Socket("127.0.0.1", 9999);  SendFileThread sendFile = new SendFileThread(frame, socket2, "Client", sourceFileChooser, status);  sendFile.start();    } catch (IOException ioException) {  ioException.printStackTrace();  }   }  });     }   @Override  public void actionPerformed(ActionEvent e) {  // get the text box's text  String text = jtf.getText();  // combine the sending messages  text = "Server: " + text;   // show the text in own window  jta.append(text + System.*lineSeparator*());  try {  // send the message  bw.write(text);  bw.newLine();  bw.flush();   // clear the text  jtf.setText("");  } catch (Exception e1) {  e1.printStackTrace();  }  }   static class FileThread extends Thread {  private int port;  public boolean flag = false;   public FileThread(int port) {  this.port = port;  }   @Override  public void run() {  Socket socket = null;  DataOutputStream dos = null;  try {  socket = new Socket("127.0.0.1", port);  System.*out*.println("Client already chose file");  DataInputStream dis = new DataInputStream(socket.getInputStream());  // get the name of file  File file = new File("/Users/calvin/"+dis.readUTF());  // get the size of file  double totleLength = dis.readLong();  dos = new DataOutputStream(new FileOutputStream(file));  // start to receive file  System.*out*.println("Starting trans: "+totleLength);  int length=-1;  byte[] buff= new byte[1024];  double curLength = 0;  while((length=dis.read(buff))>0){  dos.write(buff, 0, length);  curLength+=length;  System.*out*.println("Trans per: "+(curLength/totleLength\*100)+"%");  }  dos.flush();  System.*out*.println("Trans succeeded!");  flag = true;  } catch (IOException e) {  e.printStackTrace();  } finally {  try {  // close OutputStream  if (dos != null) {  dos.close();  }  // close socket  if (socket != null) {  socket.close();  }  } catch (IOException e) {  // *TODO Auto-generated catch block* e.printStackTrace();  }   }  }  }   static class File\_Send\_Thead extends Thread {  private ServerSocket fileSS;  private int port;   public File\_Send\_Thead(int port) {  this.port = port;  }   @Override  public void run() {  DataInputStream dis = null;  Socket socket = null;  try {  fileSS = new ServerSocket(port);  socket = fileSS.accept();  System.*out*.println("Server already chose file");  // select the file for transfer  File file = new File("/Users/calvin/IdeaProjects/test/copy-of-file.txt-2");  DataOutputStream dos = new DataOutputStream(socket.getOutputStream());  dis = new DataInputStream(new FileInputStream(file));  // get the name of file  dos.writeUTF(file.getName());  dos.flush();  // get the size of file  dos.writeLong(file.length());  dos.flush();   System.*out*.println("Starting trans(size: "+file.getTotalSpace()+")");   int length = -1;// get the size of file  byte[] buff = new byte[1024];   while ((length = dis.read(buff)) > 0) {  dos.write(buff, 0, length);  dos.flush();  }  System.*out*.println("Trans succeeded!");   } catch (IOException e) {  e.printStackTrace();  } finally {  try {  // close InputStream  if (dis != null) {  dis.close();  }  // close the socket port  if (socket != null) {  socket.close();  }  // close socket  if (fileSS != null) {  fileSS.close();  }  } catch (IOException e) {  // *TODO Auto-generated catch block* e.printStackTrace();  }  }  }  } }  客户端：  import javax.swing.\*; import java.awt.\*; import java.awt.event.ActionEvent; import java.awt.event.ActionListener; import java.io.\*; import java.net.ServerSocket; import java.net.Socket;   //1.define every blocks in frame //2.initialize all blocks public class ClientChatMain extends JFrame{  public static void main(String[] args) {  new ClientChatMain();//initialize  }   private ServerSocket serversocket;  //text area  private JTextArea jta;  //scroll pane  private JScrollPane jsp;  //blank space  private JPanel jp;  //text box  private JTextField jtf;   //button  private JButton jb;  //file send button  private JButton jsend;   // define the output stream  private BufferedWriter bw = null;    //creation  public ClientChatMain(){  //initial all block  jta = new JTextArea();  //set the text uneditable  jta.setEditable(false);  //add the text to the scroll pane to scroll  jsp = new JScrollPane(jta);  //blank space  jp = new JPanel();  //text box  jtf = new JTextField(10);  //button  jb = new JButton("Send");  //file sending button  jsend = new JButton("Send File");   //Add buttons to blank space  jp.add(jtf);  jp.add(jb);  jp.add(jsend);    //add everything into window  this.add(jsp, BorderLayout.*CENTER*); //put in the middle  this.add(jp,BorderLayout.*SOUTH*); //put in the bottom  this.add(jsend,BorderLayout.*NORTH*);   // set everything visible  this.setTitle("CTalk-Client");  this.setSize(300,300);  this.setLocation(300,300);   this.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  this.setVisible(true);   jb.addActionListener(new ActionListener() {  @Override  public void actionPerformed(ActionEvent e) {  // get the sending messages in text box  String text = jtf.getText();  String temp\_line = text;   // combine the input text  text = "Client: " + text;  // show the text in own window  if (!text.contains("[File\_request]")) jta.append(text + System.*lineSeparator*());  else {  jta.append("File Transfer Start......" + System.*lineSeparator*());  jta.append("File Name:" + text.split("#")[1] + System.*lineSeparator*());  }  try {  if (temp\_line.contains("[File\_request]")) {  int port = 9999;  FileThread filethread = new FileThread(port);  filethread.start();  System.*out*.println("File Thread start" + System.*lineSeparator*());   }  // send the message  bw.append(text);  bw.newLine();  bw.flush();   // clear the text box  jtf.setText("");  } catch (Exception e2) {  e2.printStackTrace();  }   }  });  try {  // create a server catcher  Socket socket = new Socket("127.0.0.1", 7777);  // catch input stream from socket  BufferedReader br = new BufferedReader(new InputStreamReader(socket.getInputStream()));  // catch output stream from socket  bw = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()));   // get the input data in loop, combine them into text box  String line = null;  while ((line = br.readLine()) != null) {  if (line.contains("File Transfer succeeded!")) {  jta.append("File Transfer succeeded!" + System.*lineSeparator*());  }  else jta.append(line + System.*lineSeparator*());  }   // close the socket channel  socket.close();  } catch (Exception e) {  e.printStackTrace();  }      jsend.addActionListener(new ActionListener() {  @Override  public void actionPerformed(ActionEvent e) { // System.out.println("test\_back");  JFrame frame = new JFrame();   JFileChooser sourceFileChooser = new JFileChooser(".");  sourceFileChooser.setFileSelectionMode(JFileChooser.*FILES\_ONLY*);  int status = sourceFileChooser.showOpenDialog(frame);  File sourceFile = new File(sourceFileChooser.getSelectedFile().getPath());   jta.append("Send the file: <" + sourceFile.getName() + ">\r\n");  Socket socket2 = null;  try {  socket2 = new Socket("127.0.0.1", 9999);  SendFileThread sendFile = new SendFileThread(frame, socket2, "Server", sourceFileChooser, status);  sendFile.start();    } catch (IOException ioException) {  ioException.printStackTrace();  }    }  });   }  static class FileThread extends Thread {   private ServerSocket fileSS;  private int port;   public FileThread(int port) {  this.port = port;  }   @Override  public void run() {  DataInputStream dis = null;  Socket socket = null;  try {  fileSS = new ServerSocket(port);  socket = fileSS.accept();  System.*out*.println("Client already chose file");  // select the file for transfer  File file = new File("/Users/calvin/IdeaProjects/test/copy-of-file.txt");  DataOutputStream dos = new DataOutputStream(socket.getOutputStream());  dis = new DataInputStream(new FileInputStream(file));  // get the name of file  dos.writeUTF(file.getName());  dos.flush();  // get the size of file  dos.writeLong(file.length());  dos.flush();   System.*out*.println("Starting trans(size: "+file.getTotalSpace()+")");   int length = -1;// get the size of file  byte[] buff = new byte[1024];   while ((length = dis.read(buff)) > 0) {  dos.write(buff, 0, length);  dos.flush();  }  System.*out*.println("Trans succeeded!");   } catch (IOException e) {  e.printStackTrace();  } finally {  try {  // close InputStream  if (dis != null) {  dis.close();  }  // close the socket port  if (socket != null) {  socket.close();  }  // close socket  if (fileSS != null) {  fileSS.close();  }  } catch (IOException e) {  // *TODO Auto-generated catch block* e.printStackTrace();  }   }  }  } } | | | | | | |
| 四、实验结果及分析和（或）源程序调试过程  1. 实现客户端  截屏2021-11-16 上午1.12.08   1. 实现服务端   截屏2021-11-16 上午1.12.11   1. 发送文字的GUI实现   截屏2021-11-16 上午1.12.25  截屏2021-11-16 上午1.12.29  4. 聊天对象接收文字信息  截屏2021-11-16 上午1.12.32   1. 聊天消息在服务端和客户端同步   截屏2021-11-16 上午1.12.48  截屏2021-11-16 上午1.12.58   1. 发送文件   采用手动输入 “[File\_request]#file\_name” 格式的信息开始发送文件，提醒对方开启接收文件的进程；  发送情况会反馈在GUI界面中  截屏2021-11-16 上午1.13.27  7. 文件接收  截屏2021-11-16 上午1.13.31  截屏2021-11-16 上午1.42.58   1. 文件成功接收（指定文件储存在指定目录）   截屏2021-11-16 上午1.13.48   1. 传输信息显示在终端中   截屏2021-11-16 上午1.14.02  截屏2021-11-16 上午1.14.07   1. 选定任意文件进行传输（send file按钮）   截屏2021-11-16 上午1.14.23  截屏2021-11-16 上午1.14.31 | | | | | | |

备注：

1、教师在布置需撰写实验报告的实验前，应先将报告书上的“实验题目”、“实验性质”、“实验目的”、“实验项目内容”等项目填写完成，然后再下发给学生。

2、教师在布置需撰写报告的实验项目时，应告知学生提交实验报告的最后期限。

3、学生应按照要求正确地撰写实验报告：

* 1. 在实验报告上正确地填写“实验时间”、“实验地点”等栏目。
  2. 将实验所涉及的源程序文件内容（实验操作步骤或者算法）填写在“实验过程或算法（源程序）”栏目中。
  3. 将实验所涉及源程序调试过程（输入数据和输出结果）或者实验的分析内容填写在“实验结果及分析和（或）源程序调试过程”栏目中。
  4. 在实验报告页脚的“报告创建时间：”处插入完成实验报告时的日期和时间。
  5. 学生将每个实验完成后，按实验要求的文件名通过网络提交（上载）到指定的服务器所规定的共享文件夹中。每个实验一个电子文档，如果实验中有多个电子文档（如源程序或图形等），则用WinRAR压缩成一个压缩包文档提交，压缩包文件名同实验报告文件名（见下条）。
  6. 提交的实验报告电子文档命名为：“年级（两位数字不要“级”字）专业（缩写：计算机科学与技术专业（计科）、网络工程专业（网络）、信息安全专业（信息）、物联网工程（物联网））班级（两位数字）学号（八位数字）姓名实验序号（一位数字）．doc。如学号为20115676、年级为2011级、专业为“计算机科学与技术”专业、班级为“02班”、姓名为“王宇”的学生，完成的第一次实验命名为： 11计科02班20115676王宇1．Doc，以后几次实验的报告名称以此类推。

4、教师（或助教）在评价学生实验时，应根据其提交的其他实验相关资料（例如源程序文件等）对实验报告进行仔细评价。评价后应完成的项目有:

1. 在“成绩”栏中填写实验成绩。每次实验成绩按照十分制方式评分，取整数分值。
2. 在“教师评价”栏中“评价教师签名”填写评价教师（或助教）姓名。
3. 课程实验环节结束后，任课教师将自己教学班的实验报告文件夹进行清理。在提交文件夹中，文件总数为实验次数×教学班学生人数（如，教学班人数为90人，实验项目为5，其文件数为：90×5=450）。任课教师一定要认真清理，总数相符，否则学生该实验项目不能得分。最后将学生提交的实验报告刻光盘连同实验成绩一起放入试卷袋存档。