**package** com.YanshuDu;  
  
**import** javax.swing.\*;  
*//客户端程序从这里运行***public class** TheClient **extends** JFrame {  
 **public static void** main(String[] args) {  
 **new** ClientFrame();  
 }  
}

package com.YanshuDu;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.io.DataInputStream;

import java.io.DataOutputStream;

import java.io.IOException;

import java.io.InputStream;

import java.net.Socket;

import java.util.LinkedList;

public class ClientFrame extends JFrame {

//添加Swing组件

private JTextArea textArea=new JTextArea(10,20);

private JTextField textField=new JTextField(20);

private Box box=Box.createVerticalBox();

private JButton group=new JButton("群聊（所有在线用户）");

private JLabel jl=new JLabel("在 线 用 户 列 表");

//IO

private Socket s=null;

private DataOutputStream output=null;

LinkedList<String> portList=new LinkedList<>();

//定义收件人

private String receiver="to all\n";

public ClientFrame() throws HeadlessException {

init();

}

public void init(){

this.setTitle("客户端窗口");

textArea.setEnabled(false);

this.add(box,BorderLayout.WEST);

this.add(textArea,BorderLayout.CENTER);

this.add(textField,BorderLayout.SOUTH);

box.add(group);

box.add(jl);

group.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

textArea.append("\t群聊\n");

receiver="to all\n";

}

});

textField.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

writeInTextArea(textField.getText());//在文本框中显示输出的文字，并和服务器通信

textField.setText("");

}

});

this.setBounds(300,200,600,400);

this.setVisible(true);

textField.requestFocus();//光标聚焦

//接受服务器消息

try {

s=new Socket("localhost",ServeFrame.port);

output=new DataOutputStream(s.getOutputStream());//开启输出流

} catch (IOException e) {

textArea.append("服务器未运行，请退出等待服务器开启\n");

}

textField.setToolTipText("输入文字，按回车发送");

handleInput();

this.setDefaultCloseOperation(WindowConstants.EXIT\_ON\_CLOSE);

}

private void writeInTextArea(String contain){

if(contain.equals("")) return;

textArea.append(putBlank(90-contain.length())+contain+'\n');

handleSent(contain);

}

//发送信息到服务器

public void handleSent(String str){

try {

output.writeUTF(receiver+str);//发送收件人+内容

} catch (IOException e) {

textArea.append("发送失败，您已被服务器强制下线！");

}

}

public void handleInput(){

new Thread(new Runnable() {

@Override

public void run() {

try (DataInputStream input = new DataInputStream(s.getInputStream())) {

while(true){

String in=input.readUTF();

String[] ss=in.split(" ");

if(ss[0].equals("newUser"))

{

portList.add(ss[1]);

JButton jb=new JButton("用户"+ss[1]);

box.add(jb);

jb.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

receiver="to "+ss[1]+'\n';

textArea.append("\t发送给"+ss[1]+'\n');

}

});

}else

textArea.append(in+'\n');

}

}catch (Exception e){

textArea.append("您与服务器已断开连接。");

}

}

}).start();

}

private String putBlank(int number){

StringBuilder sb=new StringBuilder();

for (int i = 0; i < number; i++) {

sb.append(" ");

}

return sb.toString();

}

}

服务器端代码

**package** com.YanshuDu;  
  
**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;  
**import** java.util.LinkedList;  
**import** java.util.Queue;  
  
**public class** ServeFrame **extends** JFrame {  
 *//Swing组件* **public static** JTextArea *serveTextArea*=**new** JTextArea();  
 **private** JPanel **jp**=**new** JPanel();  
 **private** JButton **startButton**=**new** JButton(**"启动服务器"**);  
 **private** JButton **endButton**=**new** JButton(**"终止服务器"**);  
 **private** Box **box**=Box.*createVerticalBox*();  
 **private** JLabel **jl**=**new** JLabel(**"\t\t成 员 列 表\t\t"**);  
 *//TCP连接* Socket **socket**=**null**;  
 **public static final int *port***=6666;  
 **private** DataInputStream **input**=**null**;  
 LinkedList<Handle> **list**=**new** LinkedList<>();  
 *//消息列表* Queue<String> **messageQueue**=**new** LinkedList<>();  
  
 **public static void** main(String[] args) {  
 ServeFrame serveUI=**new** ServeFrame();  
 }  
  
 **public** ServeFrame(){  
 *//界面设置* **this**.setTitle(**"Chat Room 服务器"**);  
 **this**.setBounds(0,200,600,400);  
 **this**.setDefaultCloseOperation(***EXIT\_ON\_CLOSE***);  
 **this**.setVisible(**true**);  
 **this**.setLayout(**new** BorderLayout());  
 *//加入JPanel* **this**.add(**jp**);  
 **jp**.setLayout(**new** BorderLayout());  
 *//jp中加入组件* **jp**.add(*serveTextArea*,BorderLayout.***CENTER***);  
 **jp**.add(**startButton**,BorderLayout.***SOUTH***);  
 **jp**.add(**box**,BorderLayout.***EAST***);  
 **this**.add(**endButton**,BorderLayout.***SOUTH***);  
 *serveTextArea*.setEnabled(**false**);  
 *//box属性设置* **box**.add(**jl**);  
 *//打印服务器状态  
 serveTextArea*.setFont(**new** Font(**"宋体"**, Font.***BOLD***,15));  
  
 **startButton**.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 ServeSetting();  
 **startButton**.setEnabled(**false**);  
 **endButton**.setEnabled(**true**);  
 }  
 });  
  
*// 发送消息队列,新线程* **new** Thread(**new** Runnable() {  
 @Override  
 **public void** run() {  
 **while**(**true**){  
 **while**(!**messageQueue**.isEmpty()){  
 String s=**messageQueue**.remove();  
 System.***out***.println(s);  
 String[] ss=s.split(**"\n"**);  
 **if**(ss[1].equals(**"to all"**)){  
 **for** (**var** v:**list**) {  
 *//给除了自己以外的所有人群发消息* **if**(!v.**socket**.getRemoteSocketAddress().toString().equals(ss[0])){  
 v.sentMessage(ss[0]+**"(群) : "**+ss[2]);  
 }  
 }  
 }**else** {  
 String socketAddress=ss[1].split(**" "**)[1];  
 System.***out***.println(**"to"**+socketAddress);  
 **for** (**var** v:**list**) {  
 *//给指定对象发消息* **if**(v.**socket**.getRemoteSocketAddress().toString().equals(socketAddress)){  
 v.sentMessage(ss[0]+**"(私):"**+ss[2]);  
 }  
 }  
 }  
 }  
 **try** {  
 Thread.*sleep*(1000);  
 } **catch** (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
 }).start();  
  
 **endButton**.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 *serveTextArea*.append(**"服务器已关闭\n"**);  
 **startButton**.setEnabled(**true**);  
 **endButton**.setEnabled(**false**);  
 closeServer();  
 }  
 });  
 }  
 *//关闭服务器* **public void** closeServer(){  
 System.*exit*(0);  
 }  
  
 **public void** ServeSetting() {  
 **try** {  
 ServerSocket ss=**new** ServerSocket(***port***);  
 *serveTextArea*.setText(**"服务器运行中...\n"**);  
 System.***out***.println(**"server is running..."**);  
 Thread t=**new** Thread(**new** Runnable() {  
 @Override  
 **public void** run() {  
 **try** {  
 **while**(**endButton**.isEnabled()) {*//停止服务器按钮没有被点击* **socket** = ss.accept();  
 String address=**socket**.getRemoteSocketAddress().toString();  
 *serveTextArea*.append( address+**"上线\n"** );  
 Handle tempHandle=**new** Handle(**socket**);  
 *//有新用户上线，给之前已经上线客户端发送新用户地址* **for** (**var** v:**list** ) {  
 **new** Thread(**new** Runnable() {  
 @Override  
 **public void** run() {  
 v.addMember(address);  
 }  
 }).start();  
 }  
 *//显示成员列表* displayMember(address,tempHandle);  
 **list**.add(tempHandle);  
 tempHandle.start();  
 }  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
 }  
 });  
 t.start();  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
  
 }  
 **public void** displayMember(String address,Handle handle){  
 Box Hbox=Box.*createHorizontalBox*();  
 Hbox.add(**new** JLabel(address));  
 JButton closeButton=**new** JButton(**"强制下线"**);  
 Hbox.add(closeButton);  
 **box**.add(Hbox);  
 closeButton.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 handle.interrupt();  
 closeButton.setEnabled(**false**);  
 }  
 });  
 }  
  
 **class** Handle **extends** Thread{  
 **private final** Socket **socket**;  
 DataOutputStream **writer** = **null**;  
  
 **public** Handle(Socket socket){  
 **this**.**socket**=socket;  
 }  
  
 @Override  
 **public void** run() {  
 **try**(InputStream input=**this**.**socket**.getInputStream()) {  
 **try**(OutputStream output=**this**.**socket**.getOutputStream()){  
 handle(input,output);  
 }  
 } **catch** (IOException | InterruptedException e) {  
 **try** {  
 **socket**.close();  
 } **catch** (IOException ioException) {  
 ioException.printStackTrace();  
 }  
 e.printStackTrace();  
 }  
 ServeFrame.*serveTextArea*.append(**socket**.getRemoteSocketAddress()+**" 已下线"**+**'\n'**);  
  
 }  
  
 @Override  
 **public void** interrupt() {  
 **try** {  
 **socket**.getInputStream().close();  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
 **super**.interrupt();  
 }  
  
 **private void** handle(InputStream input, OutputStream output) **throws** InterruptedException, IOException {  
 **writer** = **new** DataOutputStream(output);  
 **var** reader = **new** DataInputStream(input);  
 **try** {  
 **writer**.writeUTF(**socket**.getPort()+**"欢迎登录!\n"**);  
 *//给新上线用户发送所有之前已经存在的用户地址* sentMemberList(**writer**);  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
 **while**(!isInterrupted()) {  
 String s = reader.readUTF();  
 **messageQueue**.add(**socket**.getRemoteSocketAddress()+**"\n"**+s);*//接受客户端请求，在消息队列中加入：请求方+收件方+内容* ServeFrame.*serveTextArea*.append(**socket**.getRemoteSocketAddress()+**":"**+s+**'\n'**);*//打印SocketAddress和该客户端说的话* System.***out***.println(**socket**.toString());  
 Thread.*sleep*(1000);  
 }  
 }  
  
 **private void** sentMemberList(DataOutputStream write) **throws** IOException {  
 **for** (**int** i = 0; i < **list**.size()-1; i++) {  
 write.writeUTF(**"newUser "**+ **list**.get(i).**socket**.getRemoteSocketAddress());  
 }  
 }  
 *//有新用户上线，给之前已经上线客户端发送新用户地址* **public void** addMember(String address) {  
 **try** {  
 **writer**.writeUTF(**"newUser "**+address);  
 } **catch** (IOException e) {  
 System.***out***.println(**"传输成员地址异常"**);  
 }  
 }  
  
 **public void** sentMessage(String message){  
 **try** {  
 **writer**.writeUTF(message);  
 } **catch** (IOException e) {  
 System.***out***.println(**"消息转发失败"**);  
 }  
 }  
  
 }  
}