**《Java程序设计》**

**大作业报告**

作业名称 基于socket的网络聊天室

学生姓名 姜吕涛

学　　　　号 202192465105

学生班级 21智能科学与技术1班

作业日期 2023/12/15

作业成绩

一、问题描述

聊天室是一种通过网络连接，允许用户在同一平台上进行实时交流的应用程序。随着互联网的普及，聊天室成为人们日常生活中不可或缺的一部分。本实验旨在通过使用Java编程语言，设计和实现一个简单的聊天室应用程序，并探讨其实现过程中所涉及的关键技术和挑战。

二、设计思路

总体设计概括：

（1）通信建立模块：

采用TCP/IP协议建立通信，服务端通过创建serversocket套接字，客户端通过Socket套接字从而建立通信，建立通信后，服务端创建新的线程负责每个连接，负责数据的收发等等，客户端同样通过新的线程对数据进行接受。

（2）在线用户显示模块

客户端和服务端都通过HashMap来存储在线用户名及其socket，方便管理员对用户进行管理。每建立新的连接，客户端都会将其用户名信息以“user”，作为头部打包发送，服务端和客户端接收到信息后，将其用户名及socket存入HashMap。

（3）聊天信息模块

本设计将聊天信息主要分为系统信息，比如用户退出信息和服务端异常信息，以及普通聊天信息。在上述中提到，本设计将每个信息都加上相应头部打包发送，以便服务端和相应客户端做出不同的处理，主要头部有：系统信息sys，退出信息exit，文件传输file，聊天信息msg。

其中服务端客户端收到聊天信息会显示在主聊天框内，而系统信息和退出信息则会显示在系统信息框内。

（4）文件传输模块

在聊天的过程中，难免会出现有人想要发送文件给某一个人的情况，因此通过文件传输流实现了一对一的文件发送和接受。

（5）管理员管理用户模块

服务端管理员可以监控用户的聊天信息，如果用户有不文明语言，可以选择将其踢出聊天室。

（6）辅助功能

辅助功能包括：清空信息框、退出聊天室。

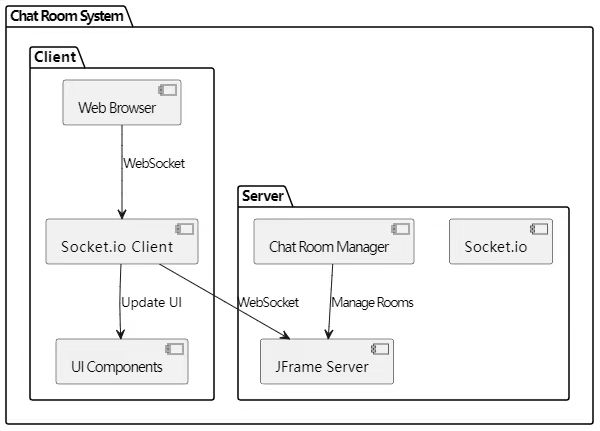
系统整体数据流图：

图1 数据流图

三、设计过程

图示

描述已自动生成客户端和服务端的程序流程图如下图所示：

图示

描述已自动生成图2 服务端程序流程图

图3 客户端程序流程图

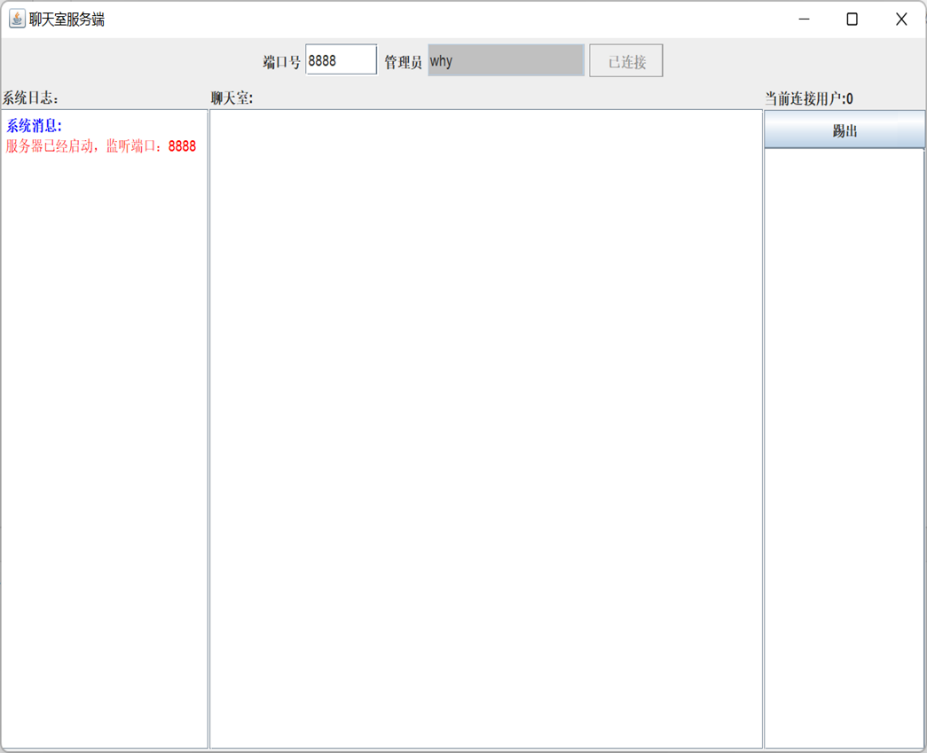
界面设计采用了Java Swing，设计了一个简易聊天室的UI界面。

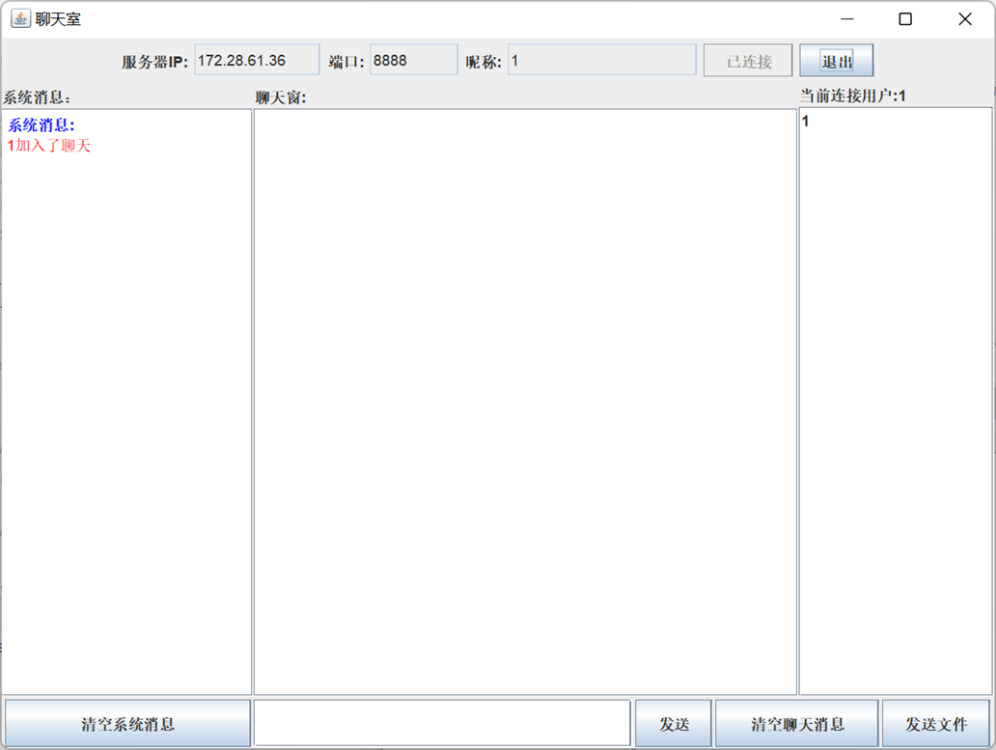
聊天信息的收发通过输入输出IO流（DataOutputStream和DataInputStream），对数据进行读写。

发送文件采用文件选择对话框 (JFileChooser)，用户选择要发送的文件，创建一个新线程来处理文件发送，以避免阻塞主线程。在新线程中，获取文件输入流 (FileInputStream) 并创建用于传输文件数据的输出流 (DataOutputStream 和 BufferedOutputStream)。向服务器发送文件相关信息，包括头部("File")、发送者名称、接收者名称、文件名和文件大小。使用循环读取文件内容并写入输出流，实现文件的传输。服务端收到文件内容并对接受客户端转发文件，实现一对一文件传输。接受客户端收到文件后，会弹出文件保存对话框，可以自由选取文件保存路径。

在设计中，实现了对IP和端口号规范性的检查算法，采用了正则表达式。

四、成果展示

服务端界面:

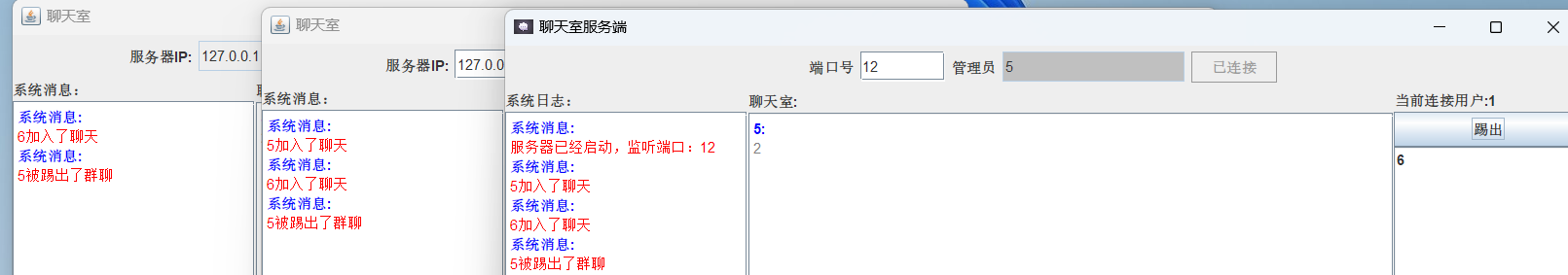
客户端界面:

电脑萤幕的截图

描述已自动生成聊天效果展示:

图形用户界面, 应用程序, 表格

描述已自动生成发送文件:

管理员踢人:

五、心得体会

本次通过设计程序，编写程序，使我对Client/Server架构的设计有了新的认识。对网络编程有了进一步的了解。在设计的过程中了解到自己的知识还是比较有限，使得程序还不够完善。回顾此次Java程序的编写。从理论到实践，可以学到很多东西，同时不仅可以巩固以前学的东西，而且学到了很多课本上学不到的东西，通过本次设计，使得自己懂得必须要把理论和实践相结合起来，从实践中得出结论，才能真正掌握这门技术，也提高了自己思考的能力。在设计的过程中，可以解决遇到的bug，解决bug的经验也是课程设计的一大收获。真正体会到要将一门知识学到可以自己运用，才是属于自己的，因此我必须加强实践，多练习。本次设计还有很多不足与缺陷，程序中还有很多问题需要完善。比如客户端登录注册并没有实现，图片的发送不能做到在线预览等等。在今后的学习中，将会不断改进和完善这方面的技能。

六、附录（关键代码）

Sever类:(服务端主线程)

package chatRoom;  
  
import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.WindowAdapter;  
import java.awt.event.WindowEvent;  
  
import java.io.IOException;  
import java.net.Socket;  
import java.util.\*;  
import java.util.List;  
  
  
*/\*\*  
 \* @brief 初始化连接 服务端的gui界面  
 \*/*public class Sever {  
 public List<Socket> onlineSockets = new ArrayList<>();  
 public HashMap<Socket,String> user\_socket = new HashMap<>();  
 /\*GUI参数\*/  
 public JFrame frame;  
  
 public JTextField port\_textfield;  
 public JTextField name\_textfield;  
 public JButton connectButton;  
  
  
  
 public JScrollPane sysTextScroll;  
 public JScrollBar sysVertical;  
 public JTextPane sysTextArea;  
  
 public DefaultListModel<String> users\_model;  
 public JList<String> userlist;  
 public JScrollPane userListPane;  
 public JButton kick;  
 public JLabel user;  
  
 public JTextPane userTextArea;  
 public JScrollPane userTextScroll;  
 public JScrollBar userVertical;  
  
  
  
  
  
  
  
 public String title;  
 private JProgressBar progressBar;  
  
 */\*\*  
 \* @MethodName GUI初始化  
 \*/* public void initview(){  
  
  
  
  
 JPanel panel = new JPanel();//主面板  
 JPanel topPanel = new JPanel(); //上面是连接按钮区域  
// JPanel bottomPanel = new JPanel();//下面发送信息区域  
 JPanel centerpanel = new JPanel(); /\*中间panel，用于放置聊天信息\*/  
 JPanel leftpanel = new JPanel(); /\*左边panel，用于放置房间列表和加入按钮\*/  
 JPanel rightpanel = new JPanel(); /\*右边panel，用于放置房间内人的列表\*/  
  
 /\*底层布局，分中间，上下左右五个部分\*/  
 BorderLayout layout = new BorderLayout();  
 /\*格子布局，主要用来设置左、右、下三个部分的布局 1\*3 \*/  
 GridBagLayout gridBagLayout = new GridBagLayout();  
 /\*主要设置上部的布局 组件按照加入的先后顺序按照设置的对齐方式从左向右排列，一行排满到下一行开始继续排列\*/  
 FlowLayout flowLayout = new FlowLayout();  
  
  
 //窗口设置在屏幕中央  
 frame = new JFrame(this.title);  
  
 frame.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
 frame.setIconImage(new ImageIcon("D:\\Java practice\\NetworkChatProject\\icon.jpg").getImage());  
  
 frame.setSize(900,600);  
 frame.setLocationRelativeTo(null);  
 frame.setContentPane(panel);  
 frame.setLayout(layout);  
  
  
  
 topPanel.setLayout(flowLayout);  
  
 leftpanel.setLayout(gridBagLayout);  
 centerpanel.setLayout(gridBagLayout);  
 rightpanel.setLayout(gridBagLayout);  
  
 leftpanel.setPreferredSize(new Dimension(200,0));  
 centerpanel.setPreferredSize(new Dimension(550,0));  
 rightpanel.setPreferredSize(new Dimension(155,0));  
  
  
 /\*------------------------------------------------头部----------------------------------------\*/  
 JLabel port\_label = new JLabel("端口号");  
 JLabel name\_label = new JLabel("管理员");  
 port\_textfield = new JTextField(); //端口号  
 port\_textfield.setPreferredSize(new Dimension(70,25));  
 name\_textfield = new JTextField(); //管理员  
 name\_textfield.setPreferredSize(new Dimension(150,25));  
  
 String name = JOptionPane.*showInputDialog*("请输入本聊天室管理员昵称：");  
 if (name != null && !name.isEmpty()) {  
 name\_textfield.setText(name);  
 }  
 name\_textfield.setEditable(false);  
 name\_textfield.setBackground(Color.*LIGHT\_GRAY*);  
 connectButton = new JButton("连接");  
  
 topPanel.add(port\_label);  
 topPanel.add(port\_textfield);  
 topPanel.add(name\_label);  
 topPanel.add(name\_textfield);  
 topPanel.add(connectButton);  
  
 /\*-------------------------------------Left----------------------------------------------------\*/  
 */\*\*  
 \* @brief 系统信息显示区域：如 xxx进入了聊天 服务器已关闭  
 \*/* JLabel sysMsg\_label = new JLabel("系统日志：");  
 sysTextArea = new JTextPane();  
 sysTextArea.setEditable(false);  
  
 sysTextScroll = new JScrollPane();//设置滚动面板  
 sysTextScroll.setViewportView(sysTextArea);//滚动面板内容  
  
 sysVertical = new JScrollBar(JScrollBar.*VERTICAL*);//滚动条  
 sysVertical.setAutoscrolls(true);  
 sysTextScroll.setVerticalScrollBar(sysVertical);//设置滚动条  
  
 leftpanel.add(sysMsg\_label,new GridBagConstraints(0, 0, 1, 1, 1, 1,  
 GridBagConstraints.*CENTER*, GridBagConstraints.*BOTH*, new Insets(0, 0, 0, 0), 0, 0));  
 leftpanel.add(sysTextScroll, new GridBagConstraints(0, 1, 1, 1, 100, 100,  
 GridBagConstraints.*CENTER*, GridBagConstraints.*BOTH*, new Insets(0, 0, 0, 0), 0, 0));  
  
  
 /\*-------------------------------------Right----------------------------------------------------\*/  
  
 user = new JLabel("当前连接用户:0");  
 kick = new JButton("踢出");  
 //管理列表的数据模型 向列表模型中添加、删除元素  
 users\_model = new DefaultListModel<>();  
 userlist = new JList<String>(users\_model);  
 userListPane = new JScrollPane(userlist);  
 rightpanel.add(user, new GridBagConstraints(0, 0, 1, 1, 1, 1,  
 GridBagConstraints.*CENTER*, GridBagConstraints.*BOTH*, new Insets(0, 0, 0, 0), 0, 0));  
 rightpanel.add(kick, new GridBagConstraints(0, 1, 1, 1, 1, 1,  
 GridBagConstraints.*CENTER*, GridBagConstraints.*BOTH*, new Insets(0, 0, 0, 0), 0, 0));  
 rightpanel.add(userListPane, new GridBagConstraints(0, 2, 1, 1, 100, 100,  
 GridBagConstraints.*CENTER*, GridBagConstraints.*BOTH*, new Insets(0, 0, 0, 0), 0, 0));  
  
 /\*-------------------------------------Center---------------------------------------------------\*/  
 JLabel userMsg\_label = new JLabel("聊天室:");  
 userTextArea = new JTextPane();  
 userTextArea.setEditable(false);  
  
 userTextScroll = new JScrollPane();  
 userTextScroll.setViewportView(userTextArea);  
  
 userVertical = new JScrollBar(JScrollBar.*VERTICAL*);  
 userVertical.setAutoscrolls(true);  
 userTextScroll.setVerticalScrollBar(userVertical);  
  
 centerpanel.add(userMsg\_label, new GridBagConstraints(0, 0, 1, 1, 1, 1,  
 GridBagConstraints.*CENTER*, GridBagConstraints.*BOTH*, new Insets(0, 0, 0, 0), 0, 0));  
 centerpanel.add(userTextScroll, new GridBagConstraints(0, 1, 1, 1, 100, 100,  
 GridBagConstraints.*CENTER*, GridBagConstraints.*BOTH*, new Insets(0, 0, 0, 0), 0, 0));  
  
  
  
  
 panel.add(topPanel, "North");  
// panel.add(bottomPanel,"South");  
 panel.add(leftpanel,"West");  
 panel.add(rightpanel,"East");  
 panel.add(centerpanel,"Center");  
  
 SeverButtonEventListener severButtonEventListener = new SeverButtonEventListener(this);  
 connectButton.addActionListener(severButtonEventListener);  
 kick.addActionListener(severButtonEventListener);  
  
 frame.addWindowListener(new WindowAdapter() {  
 @Override  
 public void windowClosing(WindowEvent e) {  
 int confirmDialog = JOptionPane.*showConfirmDialog*(frame,"确定关闭？","提示",  
 JOptionPane.*YES\_NO\_OPTION*);  
 if(confirmDialog == JOptionPane.*YES\_OPTION*){  
  
 frame.dispose();  
 System.*exit*(0);  
  
 }  
 }  
 });  
 frame.setDefaultCloseOperation(JFrame.*DO\_NOTHING\_ON\_CLOSE*);  
 frame.setVisible(true);  
 frame.repaint();  
  
  
  
 }  
  
  
  
  
  
  
 public synchronized void updateUserList(String userName) {  
 users\_model.addElement(userName);  
 userlist.revalidate();  
 userlist.repaint();  
 user.setText("当前连接用户:"+users\_model.size());  
 }  
  
  
 public void removeUserFromList(String userName) throws Exception {  
 users\_model.removeElement(userName);  
 userlist.revalidate();  
 userlist.repaint();  
 user.setText("当前连接用户:"+users\_model.size());  
 }  
  
 public void updateProgressBar(int value) {  
 progressBar.setValue(value);  
 }  
  
 public Socket findSocketByUsername(String username) {  
 Set<Map.Entry<Socket, String>> set = user\_socket.entrySet();  
 Iterator<Map.Entry<Socket, String>> iterator = set.iterator();  
  
 while (iterator.hasNext()) {  
 Map.Entry<Socket, String> entry = iterator.next();  
 if (entry.getValue().equals(username)) {  
 return entry.getKey();  
 }  
 }  
 return null;  
 }  
  
 public Sever(String title) throws IOException {  
 this.title = title;  
 }  
  
  
 public static void main(String[] args) throws IOException {  
 Sever sever = new Sever("聊天室服务端");  
 sever.initview();  
 }  
  
  
  
}

ServerThread类（服务端子线程）:

package chatRoom;  
  
import java.io.\*;  
import java.net.Socket;  
import static chatRoom.Msg2Panel.\*;  
  
  
*/\*\*  
 \* @brief 子线程 负责数据的接收 转发  
 \*/*public class SeverThread extends Thread {  
  
 public Socket clientSocket;  
 public Sever server;  
 public String Msg;  
  
  
 public SeverThread(Socket clientSocket, Sever server) {  
 this.clientSocket = clientSocket;  
 this.server = server;  
  
  
 }  
  
 @Override  
 public void run() {  
 try {  
 InputStream is = clientSocket.getInputStream();  
 DataInputStream dis = new DataInputStream(is);  
 server.onlineSockets.add(clientSocket);  
  
 while (true) {  
 try {  
  
 Msg = dis.readUTF();  
  
 int index = Msg.indexOf(",");  
 if (index!=-1) {  
 String cmd = Msg.substring(0, index);  
 String msg = Msg.substring(index + 1);  
 switch (cmd) {  
 case "user": {  
 server.user\_socket.put(clientSocket, msg);  
 handleNewUser(msg);  
  
 break;  
 }  
 case "msg": {  
 *insertMessage*(server.userTextArea, server.user\_socket.get(clientSocket) + ":", msg, false);  
 break;  
 }  
 case "File": {  
  
 String sendName = dis.readUTF();  
 String receiverName = dis.readUTF();  
  
 String fileName = dis.readUTF(); // 读取文件名  
 long fileSize = dis.readLong(); // 读取文件大小  
 System.*out*.println(fileName);  
 System.*out*.println(fileSize);  
 String filemsg = sendName + "向" + receiverName + "发送了文件：" + fileName;  
 handleSysMsg(filemsg);  
 File file = new File(fileName);  
  
 Socket socket = server.findSocketByUsername(receiverName);  
 DataOutputStream dos = new DataOutputStream(socket.getOutputStream());  
  
  
 dos.writeUTF("File," + receiverName + "/ni");  
 dos.writeUTF(fileName);  
 dos.writeLong(fileSize);  
 sendFileToClient(socket, file);  
  
 break;  
  
 }  
  
 }  
 if (!cmd.equals("File")) {  
 sendall(cmd + "," + server.user\_socket.get(clientSocket) + "/" + msg);  
 }  
 }  
  
  
 } catch (IOException e) {  
 System.*out*.println("error:"+e);  
 String name = server.user\_socket.get(clientSocket);  
 if (name != null) {  
 server.removeUserFromList(name);  
 try {  
 handleuserremove(name + "退出了群聊",name);  
 }catch (Exception f){  
 f.printStackTrace();  
 }  
 server.onlineSockets.remove(clientSocket);  
 server.user\_socket.remove(clientSocket);  
  
 dis.close();  
 clientSocket.close();  
  
  
 }  
  
  
  
 }  
 }  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 }

public void handleNewUser(String userName) throws IOException {  
  
 handleSysMsg(userName+"加入了聊天");  
 server.updateUserList(userName);  
  
 }  
 public void handleuserremove(String namemsg,String name) throws IOException {  
  
 *insertMessage*(server.sysTextArea,"系统消息:",namemsg,true);  
 sendall("exit,"+name+"/"+namemsg);  
  
 }  
 public void handleSysMsg(String msg) throws IOException {  
 *insertMessage*(server.sysTextArea,"系统消息:",msg,true);  
 sendall("sys,"+server.user\_socket.get(clientSocket)+"/"+msg);  
  
 }  
 public void sendall(String msg) throws IOException {  
 for (Socket socket : server.onlineSockets) {  
 OutputStream os = socket.getOutputStream();  
 DataOutputStream dos = new DataOutputStream(os);  
 dos.writeUTF(msg);  
 dos.flush();  
 }  
 }  
 private void sendFileToClient(Socket clientSocket,File fileToSend) throws IOException {  
 FileInputStream fis = new FileInputStream(fileToSend);  
 BufferedOutputStream bos = new BufferedOutputStream(clientSocket.getOutputStream());  
 byte[] buffer = new byte[4096];  
 int bytesRead;  
 while ((bytesRead = fis.read(buffer)) != -1) {  
 bos.write(buffer, 0, bytesRead);  
 bos.flush();  
 }  
 }  
}

ServerButton类（服务器按钮响应）:

package chatRoom;  
  
import javax.swing.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.io.IOException;  
import java.net.ServerSocket;  
import java.net.Socket;  
import static chatRoom.IP.\*;  
import static chatRoom.Msg2Panel.*insertMessage*;  
  
public class SeverButtonEventListener implements ActionListener {  
 public Sever server;  
 SeverThread severThread;  
 public SeverButtonEventListener(Sever server) {  
 this.server = server;  
 }  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 JButton source = (JButton) e.getSource();  
  
 if (source == server.connectButton) {  
 handleConnectButtonClick();  
  
 } else if (source == server.kick) {  
 try {  
 handleKickButtonClick();  
 } catch (IOException ex) {  
 throw new RuntimeException(ex);  
 }  
 }  
 }  
 private void handleConnectButtonClick() {  
 String strport;  
 int port;  
 strport = server.port\_textfield.getText();  
 if (!*ipCheckPort*(strport))  
 JOptionPane.*showMessageDialog*(server.frame, "Invalid port number", "Error", JOptionPane.*ERROR\_MESSAGE*);  
 port = Integer.*parseInt*(strport);  
 new Thread(() -> {  
 try {  
 ServerSocket serverSocket = new ServerSocket(port);  
 *insertMessage*(server.sysTextArea, "系统消息:", "服务器已经启动，监听端口：" + port, true);  
  
 server.connectButton.setEnabled(false);  
 server.connectButton.setText("已连接");  
  
 while (true) {  
 Socket clientSocket = serverSocket.accept();  
 severThread = new SeverThread(clientSocket, server);  
 severThread.start();  
  
 }  
 } catch (IOException e) {  
 JOptionPane.*showMessageDialog*(server.frame, "Failed to start the server on port " + port,  
 "Error", JOptionPane.*ERROR\_MESSAGE*);  
 }  
 }).start();  
 }  
 private void handleKickButtonClick() throws IOException {  
 String name = server.userlist.getSelectedValue(); // 获取选中的用户名  
 if(name!=null) {  
 severThread.handleuserremove(name + "被踢出了群聊",name); // 向其他用户发送系统消息  
  
 // 使用封装的方法来查找对应的 Socket  
 Socket userSocket = server.findSocketByUsername(name);  
  
 if (userSocket != null && !userSocket.isClosed()) {  
 try {  
 userSocket.close(); // 关闭找到的 Socket  
 } catch (IOException e) {  
 e.printStackTrace(); // 处理异常  
 }  
 }  
 }  
 }  
}

client类:

package chatRoom;  
  
import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.\*;  
  
  
  
public class Client {  
  
 public JFrame frame;  
 //头部参数  
 public JTextField host\_textfield;  
 public JTextField port\_textfield;  
 public JTextField name\_textfield;  
 public JButton exitButton;  
 public JButton head\_connect;  
 //底部参数  
 public JTextField text\_field;  
 public JButton foot\_send;  
 public JButton foot\_sysClear;  
 public JButton foot\_userClear;  
 public JButton foot\_sendFile;  
  
 //右边参数  
 public JLabel users\_label;  
 public JList<String> userlist;  
 public DefaultListModel<String> users\_model;  
  
 //左边参数  
 public JScrollPane sysTextScrollPane;  
 public JTextPane sysMsgArea;  
 public JScrollBar sysVertical;  
  
 //中间参数  
 public JScrollPane userTextScrollPane;  
 public JTextPane userMsgArea;  
 public JScrollBar userVertical;  
  
 private String title;  
  
  
 public Client(String title) {  
 this.title = title;  
 }  
  
  
  
  
 public void init() {  
  
  
  
  
 frame = new JFrame(this.title);  
 JPanel panel = new JPanel(); /\*主要的panel，上层放置连接区，下层放置消息区，中间是消息面板，左边是系统消息，右边是当前room的用户列表\*/  
 JPanel toppanel = new JPanel(); /\*上层panel，用于放置连接区域相关的组件\*/  
 JPanel bottompanel = new JPanel(); /\*下层panel，用于放置发送信息区域的组件\*/  
 JPanel centerpanel = new JPanel(); /\*中间panel，用于放置聊天信息\*/  
 JPanel leftpanel = new JPanel(); /\*左边panel，用于放置房间列表和加入按钮\*/  
 JPanel rightpanel = new JPanel(); /\*右边panel，用于放置房间内人的列表\*/  
  
 /\*顶层的布局，分中间，东南西北五个部分\*/  
 BorderLayout layout = new BorderLayout();  
  
 /\*格子布局，主要用来设置西、东、南三个部分的布局\*/  
 GridBagLayout gridBagLayout = new GridBagLayout();  
  
 /\*主要设置北部的布局\*/  
 FlowLayout flowLayout = new FlowLayout();  
  
 /\*设置初始窗口的一些性质\*/  
 frame.setSize(800, 600);  
 frame.setLocationRelativeTo(null);  
 frame.setDefaultCloseOperation(JFrame.*DO\_NOTHING\_ON\_CLOSE*);  
  
 frame.setContentPane(panel);  
 frame.setLayout(layout);  
  
 /\*设置各个部分的panel的布局和大小\*/  
 toppanel.setLayout(flowLayout);  
 bottompanel.setLayout(gridBagLayout);  
 leftpanel.setLayout(gridBagLayout);  
 centerpanel.setLayout(gridBagLayout);  
 rightpanel.setLayout(gridBagLayout);  
  
 //设置面板大小  
 leftpanel.setPreferredSize(new Dimension(200, 0));  
 rightpanel.setPreferredSize(new Dimension(155, 0));  
 bottompanel.setPreferredSize(new Dimension(0, 40));  
  
  
  
  
 //头部布局  
 host\_textfield = new JTextField();  
 port\_textfield = new JTextField();  
 name\_textfield = new JTextField();  
 exitButton = new JButton("退出");  
 host\_textfield.setPreferredSize(new Dimension(100, 25));  
 port\_textfield.setPreferredSize(new Dimension(70, 25));  
 name\_textfield.setPreferredSize(new Dimension(150, 25));  
  
 JLabel host\_label = new JLabel("服务器IP:");  
 JLabel port\_label = new JLabel("端口:");  
 JLabel name\_label = new JLabel("昵称:");  
  
 head\_connect = new JButton("连接");  
  
  
 toppanel.add(host\_label);  
 toppanel.add(host\_textfield);  
 toppanel.add(port\_label);  
 toppanel.add(port\_textfield);  
 toppanel.add(name\_label);  
 toppanel.add(name\_textfield);  
 toppanel.add(head\_connect);  
 toppanel.add(exitButton);  
  
 //底部布局  
 foot\_send = new JButton("发送");  
 foot\_sendFile = new JButton("发送文件");  
 foot\_sysClear = new JButton("清空系统消息");  
 foot\_sysClear.setPreferredSize(new Dimension(193, 0));  
 foot\_userClear = new JButton("清空聊天消息");  
 foot\_userClear.setPreferredSize(new Dimension(128, 0));  
 text\_field = new JTextField();  
  
 String name = JOptionPane.*showInputDialog*("请输入聊天所用昵称：");  
 if (name != null && !name.isEmpty()) {  
 name\_textfield.setText(name);  
 }  
  
 bottompanel.add(foot\_sysClear, new GridBagConstraints(0, 0, 1, 1, 1.0, 1.0,  
 GridBagConstraints.*CENTER*, GridBagConstraints.*BOTH*, new Insets(1, 1, 1, 1), 0, 0));  
 bottompanel.add(text\_field, new GridBagConstraints(1, 0, 1, 1, 100, 100,  
 GridBagConstraints.*CENTER*, GridBagConstraints.*BOTH*, new Insets(1, 1, 1, 1), 0, 0));  
 bottompanel.add(foot\_send, new GridBagConstraints(2, 0, 1, 1, 1.0, 1.0,  
 GridBagConstraints.*CENTER*, GridBagConstraints.*BOTH*, new Insets(1, 1, 1, 1), 0, 0));  
 bottompanel.add(foot\_userClear, new GridBagConstraints(3, 0, 1, 1, 1.0, 1.0,  
 GridBagConstraints.*CENTER*, GridBagConstraints.*BOTH*, new Insets(1, 1, 1, 1), 0, 0));  
 bottompanel.add(foot\_sendFile,new GridBagConstraints(4,0,1,1,1.0,1.0,  
 GridBagConstraints.*CENTER*,GridBagConstraints.*BOTH*, new Insets(1,1,1,1),0,0));  
  
 //左边布局  
 JLabel sysMsg\_label = new JLabel("系统消息：");  
 sysMsgArea = new JTextPane();  
 sysMsgArea.setEditable(false);  
 sysTextScrollPane = new JScrollPane();  
 sysTextScrollPane.setViewportView(sysMsgArea);  
 sysVertical = new JScrollBar(JScrollBar.*VERTICAL*);  
 sysVertical.setAutoscrolls(true);  
 sysTextScrollPane.setVerticalScrollBar(sysVertical);  
 leftpanel.add(sysMsg\_label, new GridBagConstraints(0, 0, 1, 1, 1, 1,  
 GridBagConstraints.*CENTER*, GridBagConstraints.*BOTH*, new Insets(0, 0, 0, 0), 0, 0));  
 leftpanel.add(sysTextScrollPane, new GridBagConstraints(0, 1, 1, 1, 100, 100,  
 GridBagConstraints.*CENTER*, GridBagConstraints.*BOTH*, new Insets(0, 0, 0, 0), 0, 0));  
  
 //右边布局  
 users\_model = new DefaultListModel<>();  
 userlist = new JList<String>(users\_model);  
 JScrollPane userListPane = new JScrollPane(userlist);  
 users\_label = new JLabel("聊天室内人数:0");  
  
  
 rightpanel.add(users\_label, new GridBagConstraints(0, 0, 1, 1, 1, 1,  
 GridBagConstraints.*CENTER*, GridBagConstraints.*BOTH*, new Insets(0, 0, 0, 0), 0, 0));  
  
  
 rightpanel.add(userListPane, new GridBagConstraints(0, 1, 1, 1, 100, 100,  
 GridBagConstraints.*CENTER*, GridBagConstraints.*BOTH*, new Insets(0, 0, 0, 0), 0, 0));  
  
 //中间布局  
 JLabel userMsg\_label = new JLabel("聊天窗:");  
 userMsgArea = new JTextPane();  
 userMsgArea.setEditable(false);  
 userTextScrollPane = new JScrollPane();  
 userTextScrollPane.setViewportView(userMsgArea);  
 userVertical = new JScrollBar(JScrollBar.*VERTICAL*);  
 userVertical.setAutoscrolls(true);  
 userTextScrollPane.setVerticalScrollBar(userVertical);  
  
 centerpanel.add(userMsg\_label, new GridBagConstraints(0, 0, 1, 1, 1, 1,  
 GridBagConstraints.*CENTER*, GridBagConstraints.*BOTH*, new Insets(0, 0, 0, 0), 0, 0));  
 centerpanel.add(userTextScrollPane, new GridBagConstraints(0, 1, 1, 1, 100, 100,  
 GridBagConstraints.*CENTER*, GridBagConstraints.*BOTH*, new Insets(0, 0, 0, 0), 0, 0));  
  
 /\*设置顶层布局\*/  
 panel.add(toppanel, "North");  
 panel.add(bottompanel, "South");  
 panel.add(leftpanel, "West");  
 panel.add(rightpanel, "East");  
 panel.add(centerpanel, "Center");  
  
  
 //将按钮事件全部注册到监听器  
 ClientButtonEventListener clientButtonEventListener = new ClientButtonEventListener(this);  
 //连接服务器  
 head\_connect.addActionListener(clientButtonEventListener);  
 foot\_send.addActionListener(clientButtonEventListener);  
 foot\_sysClear.addActionListener(clientButtonEventListener);  
 foot\_userClear.addActionListener(clientButtonEventListener);  
 foot\_sendFile.addActionListener(clientButtonEventListener);  
 exitButton.addActionListener(clientButtonEventListener);  
  
  
 //窗口关闭事件  
 frame.addWindowListener(new WindowAdapter() {  
 @Override  
 public void windowClosing(WindowEvent e) {  
 int option = JOptionPane.*showConfirmDialog*(frame, "确定关闭聊天室界面?", "提示",  
 JOptionPane.*YES\_NO\_OPTION*);  
 if (option == JOptionPane.*YES\_OPTION*) {  
 if (e.getWindow() == frame) {  
 frame.dispose();  
 System.*exit*(0);  
 }  
 }  
 }  
 });  
  
  
  
  
 //窗口显示  
 frame.setVisible(true);  
  
  
 }  
 public synchronized void updateUserList(String userName) {  
 users\_model.addElement(userName);  
 userlist.revalidate();  
 userlist.repaint();  
 users\_label.setText("当前连接用户:"+users\_model.size());  
 }  
  
 public void removeUserFromList(String name) throws Exception {  
 users\_model.removeElement(name);  
 userlist.revalidate();  
 userlist.repaint();  
 users\_label.setText("当前连接用户:"+users\_model.size());  
 }  
  
 public void removeAllUserFromList() throws Exception {  
 users\_model.removeAllElements();  
 userlist.revalidate();  
 userlist.repaint();  
 users\_label.setText("当前连接用户:"+users\_model.size());  
 }  
  
  
  
 public static void main(String[] args) throws InterruptedException {  
 Client client = new Client("聊天室");  
 client.init();  
  
 }  
}

clientReader类:

package chatRoom;  
  
import javax.swing.\*;  
import java.io.\*;  
import java.net.Socket;  
  
import static chatRoom.Msg2Panel.*insertMessage*;  
  
public class ClinentReader extends Thread {  
 private Socket socket;  
 public Client client;  
  
 public ClinentReader(Client client,Socket socket) {  
 this.socket=socket;  
 this.client = client;  
 }  
  
 @Override  
 public void run(){  
 try {  
 InputStream is = socket.getInputStream();  
 DataInputStream dis = new DataInputStream(is);  
  
 while (true) {  
 try {  
  
 String Msg = dis.readUTF();  
 System.*out*.println(Msg);  
 int index1 = Msg.indexOf(",");  
 int index2 = Msg.indexOf("/");  
  
 if(index1!=-1 && index2!=-1) {  
 String cmd = Msg.substring(0, index1);  
 String name = Msg.substring(index1 + 1, index2);  
 String msg = Msg.substring(index2 + 1);  
 switch (cmd) {  
 case "sys":  
 *insertMessage*(client.sysMsgArea, "系统消息:", msg, true);  
 break;  
 case "msg":  
 *insertMessage*(client.userMsgArea, name + ":", msg, false);  
 if(!client.users\_model.contains(name)){  
 client.updateUserList(name);  
 }  
 break;  
 case "user":{  
 client.updateUserList(name);  
 break;  
 }  
 case "exit":{  
 *insertMessage*(client.sysMsgArea, "系统消息:", msg, true);  
 client.removeUserFromList(name);  
 break;  
 }  
 case "File": {  
 String fileName = dis.readUTF();  
 long fileSize = dis.readLong();  
  
 JFileChooser fileChooser = new JFileChooser();  
 fileChooser.setDialogTitle("保存文件");  
 fileChooser.setSelectedFile(new File(fileName));  
 int userSelection = fileChooser.showSaveDialog(client.frame);  
 if (userSelection == JFileChooser.*APPROVE\_OPTION*) {  
 File fileToSave = fileChooser.getSelectedFile();  
 receiveFile(dis,fileToSave, fileSize);  
 }  
  
 }  
 }  
 }  
 } catch (IOException e) {  
 System.*out*.println("error:"+e);  
 client.removeAllUserFromList();  
 client.head\_connect.setText("连接");  
 client.port\_textfield.setEditable(true);  
 client.name\_textfield.setEditable(true);  
 client.host\_textfield.setEditable(true);  
 client.head\_connect.setEnabled(true);  
 socket.shutdownOutput();  
  
 }  
 }  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
  
  
 }  
  
 private void receiveFile(DataInputStream dis, File file, long fileSize) throws IOException {  
 try (FileOutputStream fos = new FileOutputStream(file);  
 BufferedOutputStream bos = new BufferedOutputStream(fos)) {  
  
 byte[] buffer = new byte[4096];  
 int bytesRead;  
 long totalBytesRead = 0;  
  
 while (totalBytesRead < fileSize) {  
 bytesRead = dis.read(buffer, 0, (int) Math.*min*(buffer.length, fileSize - totalBytesRead));  
 bos.write(buffer, 0, bytesRead);  
 totalBytesRead += bytesRead;  
 }  
 }  
 }  
  
}

clientButton类:

package chatRoom;  
  
import javax.swing.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.io.\*;  
import java.net.Socket;  
import java.util.Objects;  
  
  
import static chatRoom.IP.\*;  
  
public class ClientButtonEventListener implements ActionListener {  
 public Client client;  
 public DataOutputStream dos;  
 public Socket charClient;  
  
 public ClientButtonEventListener(Client client) {  
 this.client = client;  
 }  
  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 JButton source = (JButton) e.getSource();  
 if (source == client.head\_connect) {  
 handleClientConnectButtonClick();  
 } else if (source == client.foot\_send) {  
 try {  
 handleCilentSendMsgClick();  
 } catch (IOException ex) {  
 throw new RuntimeException(ex);  
 }  
 }else if (source == client.foot\_sysClear){  
 client.sysMsgArea.setText("");  
  
 } else if (source == client.foot\_userClear) {  
 client.userMsgArea.setText("");  
 }  
 else if(source == client.foot\_sendFile){  
 try {  
 handleClientSendFileClick();  
 } catch (IOException ex) {  
 ex.printStackTrace();  
 }  
 }  
 else if (source == client.exitButton){  
 try {  
 handleClientExitClick();  
 } catch (IOException ex) {  
 throw new RuntimeException(ex);  
 }  
 }  
  
 }  
  
  
 private void handleClientExitClick() throws IOException {  
 if(Objects.*equals*(client.head\_connect.getText(), "已连接")){  
 charClient.close();  
 }else{  
 JOptionPane.*showMessageDialog*(client.frame, "还没有连接服务器", "错误", JOptionPane.*ERROR\_MESSAGE*);  
 }  
  
 }  
  
 private void exitCurrentChatRoom() {  
 client.frame.dispose();  
 }  
 private void joinOtherChatRoom() {  
 Client client = new Client("聊天室");  
 client.init();  
 }  
  
  
 private void handleCilentSendMsgClick() throws IOException {  
 try {  
 String Msg = client.text\_field.getText();  
  
 sendMsg("msg,", Msg);  
 client.text\_field.setText("");  
 } catch (Exception e) {  
 JOptionPane.*showMessageDialog*(client.frame, "还没有连接服务器", "错误", JOptionPane.*ERROR\_MESSAGE*);  
 }  
 }  
  
 private void handleClientConnectButtonClick() {  
 String port;  
 String ip;  
  
 ip = client.host\_textfield.getText();  
 port = client.port\_textfield.getText();  
 if (!*ipCheckHost*(ip))  
 JOptionPane.*showMessageDialog*(client.frame, "Invalid IP address", "Error", JOptionPane.*ERROR\_MESSAGE*);  
  
 if (!*ipCheckPort*(port))  
 JOptionPane.*showMessageDialog*(client.frame, "Invalid Port", "Error", JOptionPane.*ERROR\_MESSAGE*);  
 new Thread(() -> {  
 try {  
 charClient = new Socket(ip, Integer.*parseInt*(port));  
 dos= new DataOutputStream(charClient.getOutputStream());  
 new ClinentReader(client, charClient).start();  
 client.head\_connect.setText("已连接");  
 client.port\_textfield.setEditable(false);  
 client.name\_textfield.setEditable(false);  
 client.host\_textfield.setEditable(false);  
 client.head\_connect.setEnabled(false);  
 String name = client.name\_textfield.getText();  
 sendMsg("user,", name);  
  
  
  
  
 } catch (IOException e) {  
 JOptionPane.*showMessageDialog*(client.frame, "连接服务器失败！", "错误", JOptionPane.*ERROR\_MESSAGE*);  
  
 }  
 }).start();  
  
 }  
 private void handleClientSendFileClick() throws IOException {  
 String port\_File;  
 String ip\_File;  
 String name\_Accept;  
 String name\_Send;  
 if (Objects.*equals*(client.head\_connect.getText(), "已连接")) {  
 name\_Send = client.name\_textfield.getText();  
 name\_Accept = JOptionPane.*showInputDialog*("请输入要发送的人");  
 ip\_File = client.host\_textfield.getText();  
 port\_File = client.port\_textfield.getText();  
 if (!Objects.*equals*(name\_Accept, null)) {  
 JFileChooser fileChooser = new JFileChooser();  
 fileChooser.setDialogTitle("选择文件");  
 int result = fileChooser.showOpenDialog(client.frame);  
 if (result == JFileChooser.*APPROVE\_OPTION*) {  
  
 File selectedFile = fileChooser.getSelectedFile();  
 new Thread(() -> {  
 try {  
 Socket socket = new Socket(ip\_File, Integer.*parseInt*(port\_File));  
 FileInputStream fis = new FileInputStream(selectedFile);  
 DataOutputStream dos = new DataOutputStream(socket.getOutputStream());  
 BufferedOutputStream bos = new BufferedOutputStream(dos);  
 dos.writeUTF("File," + "匿名" + "/" + "文件");  
 dos.writeUTF(name\_Send);  
 dos.writeUTF(name\_Accept);  
 dos.writeUTF(selectedFile.getName());  
 dos.writeLong(selectedFile.length());  
  
 byte[] buffer = new byte[4096];  
 int bytesRead;  
 long totalBytesRead = 0;  
  
  
 while ((bytesRead = fis.read(buffer)) != -1) {  
 bos.write(buffer, 0, bytesRead);  
 totalBytesRead += bytesRead;  
  
 }  
 } catch (IOException e) {  
 System.*out*.println("error:" + e);  
   
 }  
 }).start();  
 System.*out*.println("选择的文件路径是：" + ((File) selectedFile).getAbsolutePath());  
 }  
 } else {  
 JOptionPane.*showMessageDialog*(client.frame, "请输入要发送的人",  
 "Error", JOptionPane.*ERROR\_MESSAGE*);  
 }  
 }else{  
 JOptionPane.*showMessageDialog*(client.frame, "请先连接",  
 "Error", JOptionPane.*ERROR\_MESSAGE*);  
 }  
 }  
  
  
 public void sendMsg(String cmd, String msg) throws IOException {  
 dos.writeUTF(cmd + msg);  
 dos.flush();  
  
 }  
}

IP和端口规范检查:

package chatRoom;  
  
public class IP {  
  
 public static boolean ipCheckPort(String text){  
 return text.matches("([0-9]|[1-9]\\d{1,3}|[1-5]\\d{4}|6[0-5]{2}[0-3][0-5])");  
 }  
 public static boolean ipCheckHost(String text) {  
 if (text != null && !text.isEmpty()) {  
 // 定义正则表达式  
 String regex = "^(1\\d{2}|2[0-4]\\d|25[0-5]|[1-9]\\d|[1-9])\\."+  
 "(1\\d{2}|2[0-4]\\d|25[0-5]|[1-9]\\d|\\d)\\."+  
 "(1\\d{2}|2[0-4]\\d|25[0-5]|[1-9]\\d|\\d)\\."+  
 "(1\\d{2}|2[0-4]\\d|25[0-5]|[1-9]\\d|\\d)$";  
 // 判断ip地址是否与正则表达式匹配  
 if (text.matches(regex)) {  
 // 返回判断信息  
 return true;  
 } else {  
 // 返回判断信息  
 return false;  
 }  
 }  
 return false;  
 }  
}

信息显示在文本域:

package chatRoom;  
import javax.swing.\*;  
import javax.swing.text.\*;  
import java.awt.\*;  
  
public class Msg2Panel {  
 */\*\*  
 \* @MethodName insertMessage  
 \* @Params \* @param null  
 \* @Description 更新文本域信息格式化工具  
 \* @Return  
 \*/* public static void insertMessage( JTextPane textPane,String title, String content, boolean isSys) {  
 StyledDocument document = textPane.getStyledDocument(); /\*获取textpane中的文本\*/  
 /\*设置标题的属性\*/  
 Color content\_color = null;  
 if (isSys) {  
 content\_color = Color.*RED*;  
 } else {  
 content\_color = Color.*GRAY*;  
 }  
 SimpleAttributeSet title\_attr = new SimpleAttributeSet();  
 StyleConstants.*setBold*(title\_attr, true);  
 StyleConstants.*setForeground*(title\_attr, Color.*BLUE*);  
 /\*设置正文的属性\*/  
 SimpleAttributeSet content\_attr = new SimpleAttributeSet();  
 StyleConstants.*setBold*(content\_attr, false);  
 StyleConstants.*setForeground*(content\_attr, content\_color);  
  
  
 try {  
 document.insertString(document.getLength(), title + "\n", title\_attr);  
 document.insertString(document.getLength(), content + "\n", content\_attr);  
  
 } catch (BadLocationException ex) {  
 System.*out*.println("Bad location exception");  
 }  
 /\*设置滑动条到最后\*/  
 textPane.setCaretPosition(textPane.getDocument().getLength());  
 }  
}