

**CENTRAL SOUTH UNIVERSITY**

**Java课设实验报告**



|  |  |
| --- | --- |
| 学生姓名 |  |
| 专业班级 |  |
| 学 号 |  |
| 学 院 | 计算机学院 |
| 完成日期 | 2023.06 |

**第一章 开发工具及环境说明**

开发环境：IntelliJ IDEA Community Edition 2023.1.2、openjdk-20

**第二章 网络聊天程序业务分析**

**设计内容及要求：**

1. 用Java图形用户界面编写聊天室服务器端和客户端， 支持多个客户端连接到一个服务器。每个客户端能够输入账号。

2. 可以实现群聊（聊天记录显示在所有客户端界面）。

3. 完成好友列表在各个客户端上显示。

4. 可以实现私人聊天，用户可以选择某个其他用户，单独发送信息。

5. 服务器能够群发系统消息，能够强行让某些用户下线。

6. 客户端的上线下线要求能够在其他客户端上面实时刷新。

**第三章 网络聊天程序系统设计**

**3.1系统功能定义**

1. Main.java:创建并启动三个客户端实例
2. ChatThread.java:表示一个聊天线程，用于处理客户端和服务器之间的通信。在run方法中，根据接收到的消息类型，执行相应的操作。例如，当收到"LOGIN"消息时，将客户端的昵称添加到列表中，并更新界面；当收到"MSG"、"SMSG"或"FSMSG"消息时，将消息发送给所有用户；当收到"OFFLINE"消息时，将用户从列表中移除，并更新界面。
3. Server.java:表示服务器端的界面和逻辑。可以创建服务器的界面，并启动一个线程用于监听客户端的加入。在run方法中，接受客户端的连接，并为每个客户端创建一个ChatThread线程，然后将其加入到用户列表中。handleAll方法用于群发消息，handleExpel方法用于踢人操作。sendMessage方法用于向所有用户发送消息。
4. Client.java:实现客户端的界面和逻辑。并与服务器建立连接。在run方法中，不断读取服务器发送的消息，并根据消息类型执行相应的操作。actionPerformed方法用于处理按钮的点击事件，包括发送消息和私聊功能。并实现与外部数据库连接。

**3.2 关键问题设计**

实验中最关键的部分是服务器端和客户端的通信部分，涉及Server、ChatThread以及Client三个类。

在Server类中，通过创建ServerSocket对象并监听端口，接受客户端的连接请求。每当有客户端连接时，会创建一个新的ChatThread线程来处理与该客户端的通信。在ChatThread线程中，通过Socket对象获取输入流和输出流，实现与客户端的数据交换，根据收到的消息类型进行相应的处理，例如处理上线消息、消息发送、下线消息等。

在Client类中，首先与服务器建立连接，创建输入流和输出流。通过Socket对象与服务器进行通信，发送登录消息和消息内容，并接收服务器发送的消息。客户端也可以发送私聊消息给其他客户端，通过特定的消息格式进行私聊功能的实现。

这些部分是实验实现客户端和服务器之间通信的关键，通过其实现接收和发送消息，并根据消息的类型进行相应的处理。这些部分的正确实现和交互是确保客户端和服务器正常通信的关键。

**第四章 聊天程序源代码清单**

Main.java：

import client.Client;  
  
  
public class Main {  
  
 public static void main(String[] args) throws Exception {  
 new Client();  
 new Client();  
 new Client();  
  
 }  
}

ChatThread.java:

package server;  
  
import java.io.BufferedReader;  
import java.io.InputStreamReader;  
import java.io.PrintStream;  
import java.net.Socket;  
  
public class ChatThread extends Server implements Runnable {  
  
 Socket s = null;  
 private BufferedReader br = null;  
 public PrintStream ps = null;  
 public boolean canRun = true;  
 String nickName = null;  
  
 public ChatThread(Socket s) throws Exception {  
 this.s = s;  
 br = new BufferedReader(new InputStreamReader(s.getInputStream()));  
 ps = new PrintStream(s.getOutputStream());  
 }  
  
 public void run() {  
 while (canRun) {  
 try {  
 String msg = br.readLine();//接收客户端发来的消息  
 String[] strs = msg.split("#");  
 switch (strs[0]) {  
 case "LOGIN": //收到来自客户端的上线消息  
 nickName = strs[1];  
 dl.addElement(nickName);  
 userList.repaint();  
 sendMessage(msg);  
 break;  
 case "MSG":  
 case "SMSG":  
 case "FSMSG":  
 sendMessage(msg);  
 break;  
 case "OFFLINE": //收到来自客户端的下线消息  
 sendMessage(msg);  
 //System.out.println(msg);  
 dl.removeElement(strs[1]);  
 // 更新List列表  
 userList.repaint();  
 break;  
 }  
 } catch (Exception ignored) {  
  
 }  
 }  
 }  
}

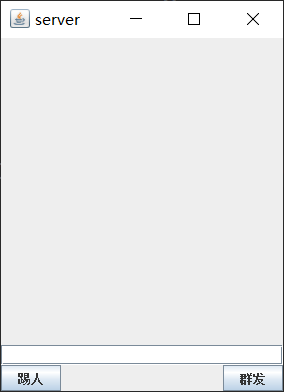
Server.java:

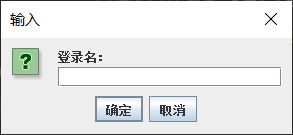
package server;  
  
import javax.swing.\*;  
import javax.swing.event.ListSelectionEvent;  
import javax.swing.event.ListSelectionListener;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
  
import java.io.BufferedReader;  
import java.io.IOException;  
import java.io.InputStreamReader;  
import java.io.PrintStream;  
import java.net.ServerSocket;  
import java.net.Socket;  
import java.util.ArrayList;  
  
class Server extends JFrame implements Runnable, ListSelectionListener, ActionListener {  
 private ServerSocket ss = null;  
 private ArrayList<ChatThread> users = new ArrayList<ChatThread>(); //容量能够动态增长的数组  
 DefaultListModel<String> dl = new DefaultListModel<String>();  
 public JList<String> userList = new JList<String>(dl);//显示对象列表并且允许用户选择一个或多个项的组件。单独的模型 ListModel 维护列表的内容。  
  
 //群发消息输入栏  
 private JTextField jtf = new JTextField();  
  
 public Server() throws Exception {  
 this.setTitle("server");  
 this.add(userList, "North");//放在北面  
 JPanel jpl = new JPanel();  
 this.add(jpl, "South");  
  
 //仅将群发消息输入栏设为一栏  
 jtf.setColumns(2);  
 jpl.setLayout(new BorderLayout());  
 jpl.add(jtf, BorderLayout.*NORTH*);  
 JButton jbt = new JButton("踢人");  
 jpl.add(jbt, BorderLayout.*WEST*);//踢出聊天室  
 JButton jbt1 = new JButton("群发");  
 jpl.add(jbt1, BorderLayout.*EAST*);//群发消息  
  
 //实现群发  
 jbt1.addActionListener(this);  
 //实现踢人  
 jbt.addActionListener(this);  
  
  
 this.setDefaultCloseOperation(*EXIT\_ON\_CLOSE*);  
 this.setLocation(400, 100);  
 this.setSize(300, 400);  
 this.setVisible(true);  
 this.setAlwaysOnTop(true);  
 ss = new ServerSocket(8888);  
 new Thread(this).start();//监听用户端的加入  
 }  
  
 @Override  
 public void run() {  
 while (true) {  
 try {  
 Socket s = ss.accept();  
 ChatThread ct = new ChatThread(s); //为该客户开一个线程  
 users.add(ct); //将每个线程加入到users  
 //发送JList里的用户登陆信息，为了防止后面登陆的用户无法更新有前面用户的好友列表  
 ListModel<String> model = userList.getModel();//获取JList的数据内容  
 for (int i = 0; i < model.getSize(); i++) {  
 ct.ps.println("USERS#" + model.getElementAt(i));  
 }  
 ct.start();  
 } catch (Exception ex) {  
 ex.printStackTrace();  
 JOptionPane.*showMessageDialog*(this, "服务器异常！");  
 System.*exit*(0);  
 }  
 }  
 }  
  
 //群发消息按钮点击事件监听  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 String label = e.getActionCommand();  
 if (label.equals("群发")) {  
 handleAll();  
 } else if (label.equals("踢人")) {  
 try {  
 handleExpel();  
 } catch (IOException e1) {  
 //e1.printStackTrace();  
 JOptionPane.*showMessageDialog*(null,"踢人失败！");  
 }  
 }  
 }  
  
 public void handleAll() {  
 if (!jtf.getText().equals("")) {  
 sendMessage("ALL#" + jtf.getText());  
 //发送完后，使输入框中内容为空  
 jtf.setText("");  
 }  
 }//群发消息  
  
 public void handleExpel() throws IOException {  
 sendMessage("OFFLINE#" + userList.getSelectedValuesList().get(0));  
 dl.removeElement(userList.getSelectedValuesList().get(0));//更新defaultModel  
 userList.repaint();//更新Jlist  
 }//踢人  
  
 class ChatThread extends Thread {  
 Socket s = null;  
 private BufferedReader br = null;  
 private PrintStream ps = null;  
 public boolean canRun = true;  
 String nickName = null;  
  
 public ChatThread(Socket s) throws Exception {  
 this.s = s;  
 br = new BufferedReader(new InputStreamReader(s.getInputStream()));  
 ps = new PrintStream(s.getOutputStream());  
 }  
  
 public void run() {  
 while (canRun) {  
 try {  
 String msg = br.readLine();//接收客户端发来的消息  
 String[] strs = msg.split("#");  
 switch (strs[0]) {  
 case "LOGIN": //收到来自客户端的上线消息  
 nickName = strs[1];  
 dl.addElement(nickName);  
 userList.repaint();  
 sendMessage(msg);  
 break;  
 case "MSG":  
 case "SMSG":  
 case "FSMSG":  
 sendMessage(msg);  
 break;  
 case "OFFLINE": //收到来自客户端的下线消息  
 sendMessage(msg);  
 //System.out.println(msg);  
 dl.removeElement(strs[1]);  
 // 更新List列表  
 userList.repaint();  
 break;  
 }  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 }  
 }  
 }  
  
 public void sendMessage(String msg) { //服务器端发送给所有用户  
 for (ChatThread ct : users) {  
 ct.ps.println(msg);  
 }  
 }  
  
 @Override  
 public void valueChanged(ListSelectionEvent e) {  
 // *TODO 自动生成的方法存根* }  
  
 public static void main(String[] args) throws Exception {  
 new Server();  
 }  
}

Client.java:

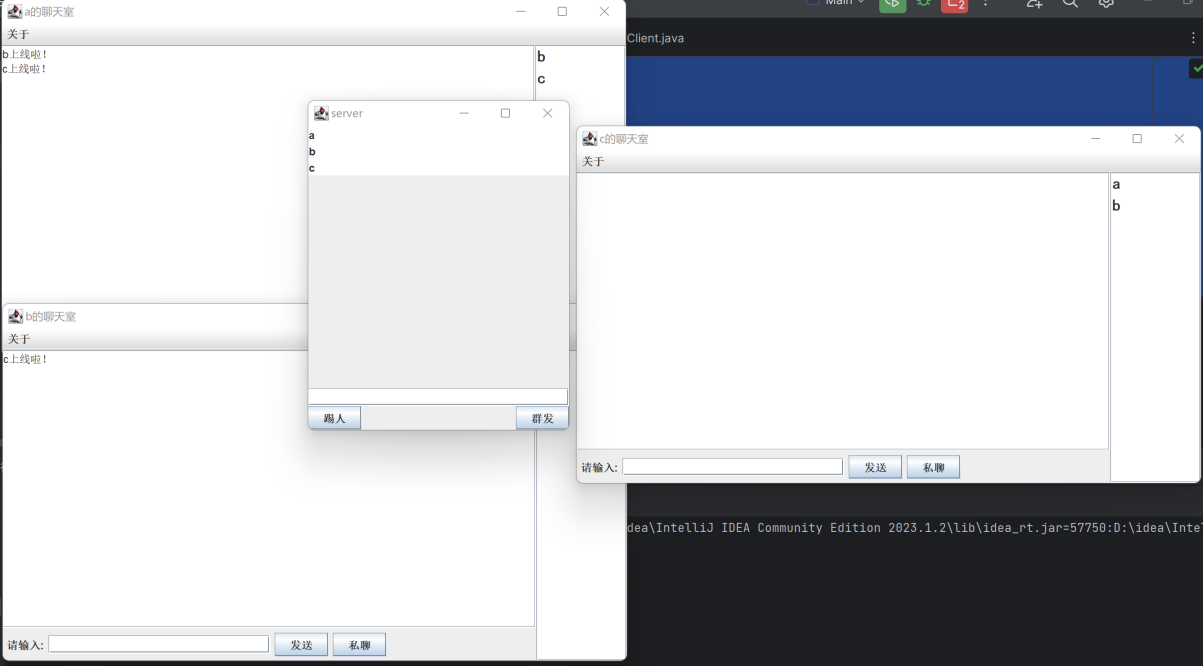
package client;  
  
import java.awt.BorderLayout;  
import java.awt.Component;  
import java.awt.Dimension;  
import java.awt.FlowLayout;  
import java.awt.Font;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.awt.event.ComponentAdapter;  
import java.awt.event.ComponentEvent;  
import java.awt.event.KeyAdapter;  
import java.awt.event.KeyEvent;  
import java.awt.event.WindowAdapter;  
import java.awt.event.WindowEvent;  
import java.io.BufferedReader;  
import java.io.InputStreamReader;  
import java.io.PrintStream;  
import java.net.Socket;  
  
import javax.swing.DefaultListModel;  
import javax.swing.JButton;  
import javax.swing.JFrame;  
import javax.swing.JLabel;  
import javax.swing.JList;  
import javax.swing.JMenu;  
import javax.swing.JMenuBar;  
import javax.swing.JMenuItem;  
import javax.swing.JOptionPane;  
import javax.swing.JPanel;  
import javax.swing.JScrollPane;  
import javax.swing.JTextArea;  
import javax.swing.JTextField;  
  
import java.sql.Connection;  
import java.sql.DriverManager;  
import java.sql.PreparedStatement;  
  
public class Client extends JFrame implements Runnable, ActionListener {  
 JPanel north = new JPanel();  
 //west  
 JPanel west = new JPanel();  
 DefaultListModel<String> dl = new DefaultListModel<String>();//用来修改JList  
 private JList<String> userList = new JList<String>(dl);//用来展示和选择  
 JScrollPane listPane = new JScrollPane(userList);  
 //center  
 JPanel center = new JPanel();  
 JTextArea jta = new JTextArea(10, 20);  
 JScrollPane js = new JScrollPane(jta);  
 JPanel operPane = new JPanel();//发送消息的操作面板  
 JLabel input = new JLabel("请输入:");  
 JTextField jtf = new JTextField(24);  
  
 JButton jButton = new JButton("发消息");  
  
 private BufferedReader br = null;  
 private PrintStream ps = null;  
 private String nickName = null;  
  
 //私聊面板  
 JTextArea jTextArea = new JTextArea(11, 40);  
 JScrollPane js1 = new JScrollPane(jTextArea);  
 JTextField jTextField = new JTextField(25);  
 String suser = "";  
 double MAIN\_FRAME\_LOC\_X;//父窗口x坐标  
 double MAIN\_FRAME\_LOC\_Y;//父窗口y坐标  
  
 boolean FirstSecret = true;//是否第一次私聊  
 String sender = null;//私聊发送者的名字  
 String receiver = null;//私聊接收者的名字  
 private Connection connection;  
 private PreparedStatement insertStatement;  
  
 public Client() throws Exception {  
  
 // Connect to MySQL database  
 String url = "jdbc:mysql://localhost:3306/chat";  
 String username = "root";  
 String password = "123456";  
 connection = DriverManager.*getConnection*(url, username, password);  
  
 // Prepare the SQL statement for inserting records  
 String insertQuery = "INSERT INTO chat\_records (sender, message) VALUES (?, ?)";  
 insertStatement = connection.prepareStatement(insertQuery);  
 Socket s = null;  
 //north 菜单栏  
 //north  
 JMenuBar bar = new JMenuBar();  
 JMenu menu = new JMenu("关于");  
 bar.add(menu);  
 JMenuItem about = new JMenuItem("关于本软件");  
 menu.add(about);  
 JMenuItem exit = new JMenuItem("退出");  
 menu.add(exit);  
 about.addActionListener(this);  
 exit.addActionListener(this);  
 BorderLayout bl = new BorderLayout();  
 north.setLayout(bl);  
 north.add(bar, BorderLayout.*NORTH*);  
 add(north, BorderLayout.*NORTH*);  
  
 //east 好友列表  
 Dimension dim = new Dimension(100, 150);  
 west.setPreferredSize(dim);//在使用了布局管理器后用setPreferredSize来设置窗口大小  
 //Dimension dim2 = new Dimension(100,150);  
 //listPane.setPreferredSize(dim2);  
 BorderLayout bl2 = new BorderLayout();  
 west.setLayout(bl2);  
 west.add(listPane, BorderLayout.*CENTER*);//显示好友列表  
 add(west, BorderLayout.*EAST*);  
 userList.setFont(new Font("楷体", Font.*BOLD*, 18));  
  
 //center 聊天消息框 发送消息操作面板  
 jta.setEditable(false);//消息显示框是不能编辑的  
 jTextArea.setEditable(false);  
  
 BorderLayout bl3 = new BorderLayout();  
 center.setLayout(bl3);  
 FlowLayout fl = new FlowLayout(FlowLayout.*LEFT*);  
 operPane.setLayout(fl);  
 operPane.add(input);  
 operPane.add(jtf);  
 JButton jbt = new JButton("发送");  
 operPane.add(jbt);  
 JButton jbt1 = new JButton("私聊");  
 operPane.add(jbt1);  
 center.add(js, BorderLayout.*CENTER*);//js是消息展示框JScrollPane  
 center.add(operPane, BorderLayout.*SOUTH*);  
 add(center, BorderLayout.*CENTER*);  
  
 js.setVerticalScrollBarPolicy(JScrollPane.*VERTICAL\_SCROLLBAR\_AS\_NEEDED*);//需要时才显示滚动条  
  
 //鼠标事件，点击  
 jbt.addActionListener(this);  
 jbt1.addActionListener(this);  
 this.setDefaultCloseOperation(*DISPOSE\_ON\_CLOSE*);  
 //this.setAlwaysOnTop(true);  
  
 nickName = JOptionPane.*showInputDialog*("登录名：");  
 this.setTitle(nickName + "的聊天室");  
 this.setSize(700, 400);  
 this.setVisible(true);  
  
 try{  
 s = new Socket("127.0.0.1", 8888);  
 br = new BufferedReader(new InputStreamReader(s.getInputStream()));  
 ps = new PrintStream(s.getOutputStream());  
 } catch (Exception e) {  
 JOptionPane.*showMessageDialog*(null,"连接服务器失败");  
 assert false;  
  
 }  
  
  
 new Thread(this).start();//run()  
 ps.println("LOGIN#" + nickName);//发送登录信息，消息格式：LOGIN#nickName  
  
 jtf.setFocusable(true);//设置焦点  
  
 //键盘事件，实现当输完要发送的内容后，直接按回车键，实现发送  
 //监听键盘相应的控件必须是获得焦点（focus）的情况下才能起作用  
 jtf.addKeyListener(new KeyAdapter() {  
 @Override  
 public void keyPressed(KeyEvent e) {  
 if (e.getKeyCode() == KeyEvent.*VK\_ENTER*) {  
 ps.println("MSG#" + nickName + "#" + jtf.getText());//发送消息的格式：MSG#nickName#message  
 //发送完后，是输入框中内容为空  
 jtf.setText("");  
 }  
 }  
 });  
  
 //私聊消息框按回车发送消息  
 jTextField.addKeyListener(new KeyAdapter() {  
 @Override  
 public void keyPressed(KeyEvent e) {  
 if (e.getKeyCode() == KeyEvent.*VK\_ENTER*) {  
 handleSS();  
 }  
 }  
 });  
  
 //监听系统关闭事件，退出时给服务器端发出指定消息  
 this.addWindowListener(new WindowAdapter() {  
 @Override  
 public void windowClosing(WindowEvent e) {  
 ps.println("OFFLINE#" + nickName);//发送下线信息，消息格式：OFFLINE#nickName  
 }  
 });  
  
 this.addComponentListener(new ComponentAdapter() {//监听父窗口大小的改变  
 public void componentMoved(ComponentEvent e) {  
 Component comp = e.getComponent();  
 MAIN\_FRAME\_LOC\_X = comp.getX();  
 MAIN\_FRAME\_LOC\_Y = comp.getY();  
 }  
 });  
 }  
  
 public void run() {//客户端与服务器端发消息的线程  
 while (true) {  
 try {  
 String msg = br.readLine();//读取服务器是否发送了消息给该客户端  
 String[] strs = msg.split("#");  
 //判断是否为服务器发来的登陆信息  
 if (strs[0].equals("LOGIN")) {  
 if (!strs[1].equals(nickName)) {//不是本人的上线消息就显示，本人的不显示  
 jta.append(strs[1] + "上线啦！\n");  
 dl.addElement(strs[1]);//DefaultListModel来更改JList的内容  
 userList.repaint();  
 }  
 } else if (strs[0].equals("MSG")) {//接到服务器发送消息的信息  
 if (!strs[1].equals(nickName)) {//别人说的  
 jta.append(strs[1] + "说：" + strs[2] + "\n");  
 } else {  
 jta.append("我说：" + strs[2] + "\n");  
 }  
 } else if (strs[0].equals("USERS")) {//USER消息，为新建立的客户端更新好友列表  
 dl.addElement(strs[1]);  
 userList.repaint();  
 } else if (strs[0].equals("ALL")) {  
 jta.append("系统消息：" + strs[1] + "\n");  
 } else if (strs[0].equals("OFFLINE")) {  
 if (strs[1].equals(nickName)) {//如果是自己下线的消息，说明被服务器端踢出聊天室，强制下线  
 JOptionPane.*showMessageDialog*(this, "您已被系统请出聊天室！");  
 System.*exit*(0);  
 }  
 jta.append(strs[1] + "下线啦！\n");  
 dl.removeElement(strs[1]);  
 userList.repaint();  
 } else if ((strs[2].equals(nickName) || strs[1].equals(nickName)) && strs[0].equals("SMSG")) {  
 if (!strs[1].equals(nickName)) {  
 jTextArea.append(strs[1] + "说：" + strs[3] + "\n");  
 jta.append("系统提示：" + strs[1] + "私信了你" + "\n");  
 } else {  
 jTextArea.append("我说：" + strs[3] + "\n");  
 }  
 } else if ((strs[2].equals(nickName) || strs[1].equals(nickName)) && strs[0].equals("FSMSG")) {  
 sender = strs[1];  
 receiver = strs[2];  
 //接收方第一次收到私聊消息，自动弹出私聊窗口  
 if (!strs[1].equals(nickName)) {  
 FirstSecret = false;  
 jTextArea.append(strs[1] + "说：" + strs[3] + "\n");  
 jta.append("系统提示：" + strs[1] + "私信了你" + "\n");  
 handleSec(strs[1]);  
 } else {  
 jTextArea.append("我说：" + strs[3] + "\n");  
 }  
 }  
 } catch (Exception ex) {//如果服务器端出现问题，则客户端强制下线  
 JOptionPane.*showMessageDialog*(this, "您已被系统请出聊天室！");  
 System.*exit*(0);  
 }  
 }  
 }  
  
  
 @Override  
 public void actionPerformed(ActionEvent e) {//鼠标点击事件  
 String label = e.getActionCommand();  
 if (label.equals("发送")) {//群发  
 handleSend();  
 } else if (label.equals("私聊") && !userList.isSelectionEmpty()) {//未点击用户不执行  
 suser = userList.getSelectedValuesList().get(0);//获得被选择的用户  
 handleSec(suser);//创建私聊窗口  
 sender = nickName;  
 receiver = suser;  
 } else if (label.equals("发消息")) {  
 handleSS();//私发消息  
 } else if (label.equals("关于本软件")) {  
 JOptionPane.*showMessageDialog*(this, "1.可以在聊天框中进行群聊\n\n2.可以点击选择用户进行私聊");  
 } else if (label.equals("退出")) {  
 JOptionPane.*showMessageDialog*(this, "您已成功退出！");  
 ps.println("OFFLINE#" + nickName);  
 System.*exit*(0);  
 } else {  
 System.*out*.println("不识别的事件");  
 }  
 }  
  
 public void handleSS() {//在私聊窗口中发消息  
 String name = sender;  
 if (sender.equals(nickName)) {  
 name = receiver;  
 }  
 if (FirstSecret) {  
 ps.println("FSMSG#" + nickName + "#" + name + "#" + jTextField.getText());  
 jTextField.setText("");  
 FirstSecret = false;  
 } else {  
 ps.println("SMSG#" + nickName + "#" + name + "#" + jTextField.getText());  
 jTextField.setText("");  
 }  
 }  
  
 public void handleSend() {//群发消息  
 //发送信息时标识一下来源  
 ps.println("MSG#" + nickName + "#" + jtf.getText());  
 //发送完后，是输入框中内容为空  
 jtf.setText("");  
 }  
  
 public void handleSec(String name) { //建立私聊窗口  
 setDefaultCloseOperation(*DISPOSE\_ON\_CLOSE*);  
 JFrame jFrame = new JFrame();//新建了一个窗口  
 JPanel JPL = new JPanel();  
 JPanel JPL2 = new JPanel();  
 FlowLayout f2 = new FlowLayout(FlowLayout.*LEFT*);  
 JPL.setLayout(f2);  
 JPL.add(jTextField);  
 JPL.add(jButton);  
 JPL2.add(js1, BorderLayout.*CENTER*);  
 JPL2.add(JPL, BorderLayout.*SOUTH*);  
 jFrame.add(JPL2);  
  
 jButton.addActionListener(this);  
 jTextArea.setFont(new Font("宋体", Font.*PLAIN*, 15));  
 jFrame.setSize(400, 310);  
 jFrame.setLocation((int) MAIN\_FRAME\_LOC\_X + 20, (int) MAIN\_FRAME\_LOC\_Y + 20);//将私聊窗口设置总是在父窗口的中间弹出  
 jFrame.setTitle("与" + name + "正在私聊");  
 jFrame.setVisible(true);  
  
 jTextField.setFocusable(true);//设置焦点  
  
 jFrame.addWindowListener(new WindowAdapter() {  
 @Override  
 public void windowClosing(WindowEvent e) {  
 jTextArea.setText("");  
 FirstSecret = true;  
 }  
 });  
 }//私聊窗口  
  
  
  
}

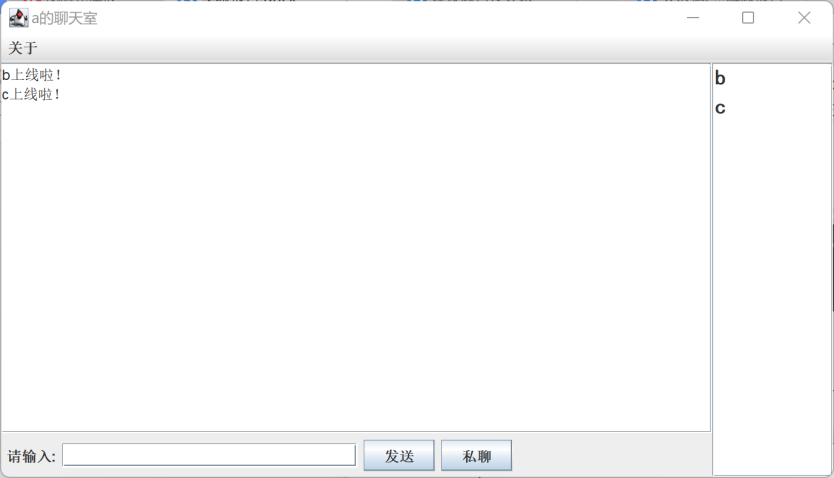
**第五章 聊天程序运行结果与测试分析**

**多个客户端连接到一个服务器。每个客户端能够输入账号：**

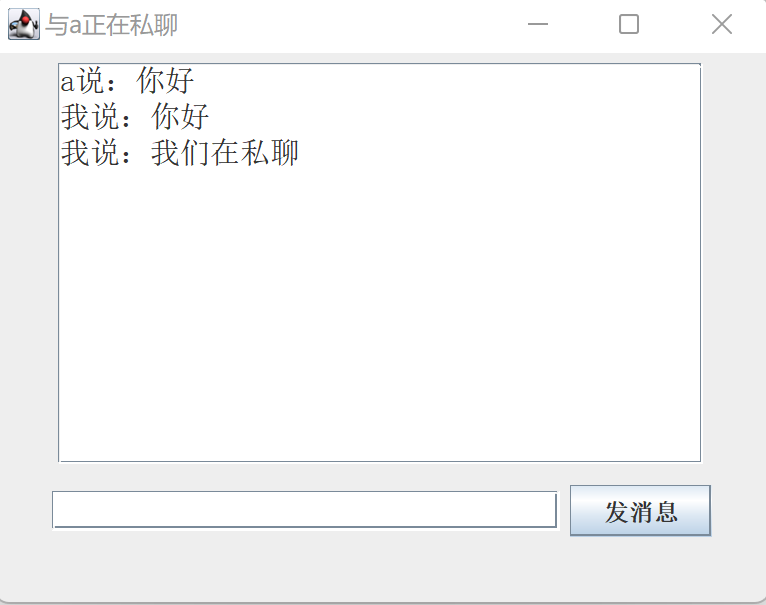
****

**用户登录后的界面：**

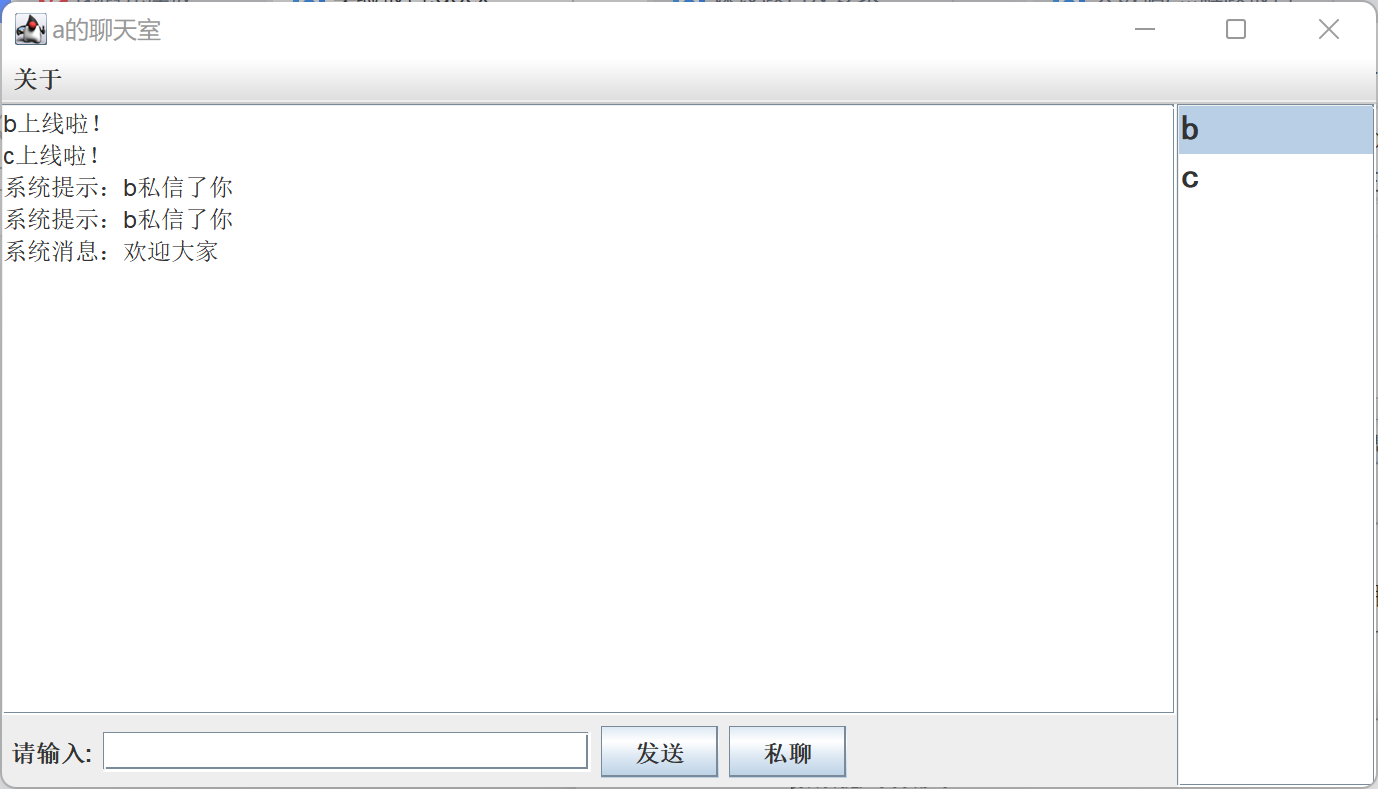


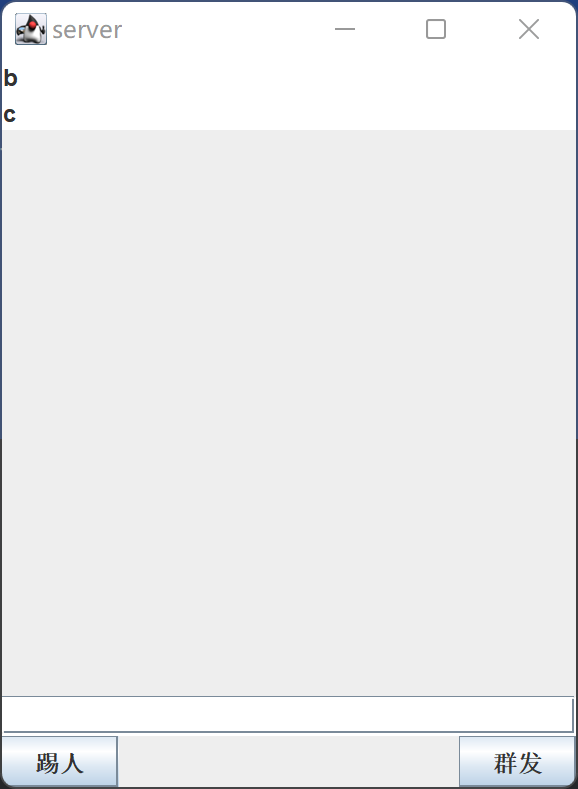
**好友列表在各个客户端上显示：**

实现私人聊天，用户可以选择某个其他用户，单独发送信息：



**服务器能够群发系统消息，能够强行让某些用户下线；客户端的上线下线要求能够在其他客户端上面实时刷新：**





1. **用JavaURL编程爬取并分析网页敏感词 业务分析**

1.编写界面，输入一个网址，能够爬取该网址上所有的HTML源代码。

2.对网址中的文本进行提取。

3.建立敏感词库，用文本文件保存。

4.将该网址所对应的文本中的敏感词提取并高亮显示。

5.编写文本文件，可以存入多个网址；程序可爬取这些网址中的文本内容，将敏感词记录存入另一个文件，格式自定。

6.编写一个主界面,整合上述功能。

**第七章 用JavaURL编程爬取并分析网页敏感词 系统设计**

**7.1系统功能定义**

1. Spider.java:使用Java Swing创建一个界面运行简单的爬虫程序，可以实现用于从指定网址中获取HTML源代码，并在源代码中匹配敏感词。
2. Main.java:实现方法，开始爬取。
3. KeywordAnalysis.java:读取一个文本文件data.txt，提取关键词并统计其出现次数，然后将结果保存到data.csv文件。
4. MySQLImportCSV.java:利用Java的JDBC API建立数据库连接，将CSV文件中的数据导入到本地的MySQL数据库。

**7.2 关键问题设计**

**1.爬取网页的功能：**

getHtml(String website) 方法使用URL爬取指定网址的HTML代码，并返回HTML代码的文本。

SpiderOne 类是一个线程，用于在后台运行爬取网页的操作。它调用了getHtml方法获取网页的HTML代码，并将代码显示在界面上。

**2.敏感词匹配的功能：**

getText(String str) 方法使用正则表达式匹配HTML代码，提取其中的文本内容，并返回纯文本。

getLib() 方法用于从文件中读取敏感词，并将敏感词添加到wordList中。

showSensword() 方法使用正则表达式匹配文本中的敏感词，并在文本框中进行高亮显示。

match 按钮的事件处理方法调用了showSensword()方法，实现了匹配敏感词并高亮显示的功能。

这些关键部分组合在一起，用户可以通过输入网址进行网页爬取，并在爬取的网页文本中匹配敏感词并进行高亮显示。

**第八章用JavaURL编程爬取并分析网页敏感词源代码清单**

1. Spider.java:

import javax.swing.\*;  
import javax.swing.text.BadLocationException;  
import javax.swing.text.DefaultHighlighter.DefaultHighlightPainter;  
import javax.swing.text.Highlighter;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.io.\*;  
import java.net.URL;  
import java.net.URLConnection;  
import java.nio.charset.StandardCharsets;  
import java.util.ArrayList;  
import java.util.regex.Matcher;  
import java.util.regex.Pattern;  
  
public class Spider extends JFrame implements ActionListener {  
  
 private JTextField siteField = new JTextField(25);  
 private JButton goSpider = new JButton("开始爬取");  
 private JTextArea htmlArea = new JTextArea(15, 25);  
 private JTextArea textArea = new JTextArea(15, 25);  
 private JTextArea sensWord = new JTextArea(8, 25);  
 private JButton openLib = new JButton(" 导入敏感词库");  
 private JButton match = new JButton("匹配");  
 private JButton siteLib = new JButton("导入网址库");  
  
 private ArrayList<String> wordList = new ArrayList<String>(); //保存敏感词  
 private ArrayList<Integer> wordNum = new ArrayList<Integer>(); //保存对应敏感词的出现次数  
 //设置正则表达式的匹配符  
 /\*regExHtml：用于匹配HTML标签（<tag>）的模式。  
 regExScript：用于匹配<script>标签及其内容的模式。  
 regExStyle：用于匹配<style>标签及其内容的模式。  
 regExSpace：用于匹配连续空格或回车等的模式。  
 regExImg：用于匹配网页上图案的乱码的模式。\*/  
 private String regExHtml = "<[^>]+>";  
 private String regExScript = "<script[^>]\*?>[\\s\\S]\*?<\\/script>";  
 private String regExStyle = "<style[^>]\*?>[\\s\\S]\*?<\\/style>";  
 private String regExSpace = "[\\s]{2,}";  
 private String regExImg = "&[\\S]\*?;+";  
 //定义正则表达式  
 private Pattern pattern1 = Pattern.*compile*(regExScript, Pattern.*CASE\_INSENSITIVE*);  
 private Pattern pattern2 = Pattern.*compile*(regExStyle, Pattern.*CASE\_INSENSITIVE*);  
 private Pattern pattern3 = Pattern.*compile*(regExHtml, Pattern.*CASE\_INSENSITIVE*);  
 private Pattern pattern4 = Pattern.*compile*(regExSpace, Pattern.*CASE\_INSENSITIVE*);  
 private Pattern pattern5 = Pattern.*compile*(regExImg, Pattern.*CASE\_INSENSITIVE*);  
  
 public Spider() throws IOException {  
 //设置界面风格  
 try {  
 UIManager.*setLookAndFeel*(UIManager.*getSystemLookAndFeelClassName*());  
 } catch (ClassNotFoundException | InstantiationException | UnsupportedLookAndFeelException | IllegalAccessException e1) {  
 // *TODO Auto-generated catch block* e1.printStackTrace();  
 }  
  
 this.setTitle("Spider");  
 this.setLocation(400, 200);  
 this.setSize(600, 500);  
 this.setDefaultCloseOperation(*EXIT\_ON\_CLOSE*);  
 JPanel jPanel = new JPanel();  
 jPanel.setLayout(new BorderLayout());  
  
 //界面处理，提醒输入网址,爬取按钮,以及编码方式选择  
 JPanel jpl1 = new JPanel();  
 jpl1.setLayout(new BorderLayout());  
 JLabel siteWarn = new JLabel("输入网址:");  
 siteWarn.setPreferredSize(new Dimension(70, 30));  
 JScrollPane siteSPane = new JScrollPane(siteField);  
 siteSPane.setPreferredSize(new Dimension(300, 30));  
 goSpider.setPreferredSize(new Dimension(90, 30));  
 JPanel jpl5 = new JPanel();  
 jpl5.setLayout(new GridLayout(1, 2, 10, 10));  
 jpl5.add(goSpider);  
 //jpl5.add(charset);  
 jpl1.add(siteWarn, BorderLayout.*WEST*);  
 jpl1.add(siteSPane, BorderLayout.*CENTER*);  
 jpl1.add(jpl5, BorderLayout.*EAST*);  
 //源代码文本,以及处理后的文本框设置  
 htmlArea.setEditable(false);  
 htmlArea.setLineWrap(true);  
 htmlArea.setFont(new Font("宋体", Font.*PLAIN*, 14));  
 JPanel jpl2 = new JPanel();  
 jpl2.setLayout(new BorderLayout());  
 JScrollPane htmlSPane = new JScrollPane(htmlArea);  
 jpl2.add(htmlSPane, BorderLayout.*CENTER*);  
 //设置布局  
 JPanel jpl8 = new JPanel();  
 jpl8.setLayout(new GridLayout(2, 1, 10, 5));  
 jpl8.add(siteLib);  
 jpl8.add(openLib);  
  
 JPanel jpl3 = new JPanel();  
 jpl3.setLayout(new BorderLayout());  
 sensWord.setLineWrap(true);  
 sensWord.setEditable(false);  
 JScrollPane wordPane = new JScrollPane(sensWord);  
 wordPane.setPreferredSize(new Dimension(6, 400));  
 jpl3.add(jpl8, BorderLayout.*NORTH*);  
 jpl3.add(wordPane, BorderLayout.*CENTER*);  
 jpl3.add(match, BorderLayout.*SOUTH*);  
  
 textArea.setFont(new Font("宋体", Font.*PLAIN*, 14));  
 textArea.setEditable(false);  
 textArea.setLineWrap(true);  
 JPanel jpl4 = new JPanel();  
 jpl4.setLayout(new BorderLayout());  
 JScrollPane textSPane = new JScrollPane(textArea);  
 jpl4.add(textSPane, BorderLayout.*CENTER*);  
  
 JTabbedPane tabPane = new JTabbedPane();  
 tabPane.add("html源代码", jpl2);  
 tabPane.add("网页文本", jpl4);  
 JPanel jpl7 = new JPanel();  
 jpl7.setLayout(new BorderLayout());  
 jpl7.add(tabPane, BorderLayout.*CENTER*);  
  
 JPanel jpl6 = new JPanel();  
 jpl6.setLayout(new BorderLayout());  
 jpl6.add(jpl7, BorderLayout.*CENTER*);  
 jpl6.add(jpl3, BorderLayout.*EAST*);  
  
 jPanel.add(jpl1, BorderLayout.*NORTH*);  
 jPanel.add(jpl6, BorderLayout.*CENTER*);  
 this.add(jPanel);  
 this.setVisible(true);  
  
 //事件处理  
 goSpider.addActionListener(this);  
 siteLib.addActionListener(this);  
 openLib.addActionListener(this);  
 match.addActionListener(this);  
 }  
  
 //使用URL爬取网页的html代码  
 public String getHtml(String website) {  
  
 String str = null;  
 String text = ""; //保存网页的内容  
 try {  
 URL url = new URL(website); //建立对应的URL对象  
 URLConnection urlConn = url.openConnection(); //连接  
 urlConn.connect();  
 //获取输入流  
 BufferedReader br = new BufferedReader(new InputStreamReader(urlConn.getInputStream(), StandardCharsets.*UTF\_8*));  
 while (true) { //爬取到结束  
 str = br.readLine();  
 if (str == null) break;  
 text += (str + "\n");  
 }  
 br.close(); //关闭输入流  
 } catch (Exception e) {  
 // *TODO: handle exception* JOptionPane.*showMessageDialog*(null, website + "爬取源代码失败");  
 }  
 //System.out.println("爬取结束");  
 return text; //返回html代码文本  
 }  
  
 //对html进行正则匹配,提取出其中的文本  
 public String getText(String str) {  
 Matcher matcher = pattern1.matcher(str);  
 str = matcher.replaceAll(""); //匹配script标签  
 System.*out*.println(str);  
 matcher = pattern2.matcher(str);  
 str = matcher.replaceAll(""); //匹配style标签  
 System.*out*.println(str);  
 matcher = pattern3.matcher(str);  
 str = matcher.replaceAll(""); //匹配普通标签  
 System.*out*.println(str);  
 matcher = pattern4.matcher(str);  
 str = matcher.replaceAll("\n"); //匹配连续回车或空格  
 System.*out*.println(str);  
 matcher = pattern5.matcher(str);  
 str = matcher.replaceAll(""); //匹配网页图案出现的乱码  
 System.*out*.println(str);  
 return str; //返回文本  
 }  
  
 //从文件中读取敏感词  
 public void getLib() {  
 JFileChooser fChooser = new JFileChooser(); //文件选择框  
 int ok = fChooser.showOpenDialog(this);  
 if (ok != JFileChooser.*APPROVE\_OPTION*) return; //判断是否正常选择  
 wordList.clear(); //清空之前的记录  
 sensWord.setText("");  
 File choosenLib = fChooser.getSelectedFile(); //获取选择的文件  
 BufferedReader br = null;  
 try { //读取选中文件中的记录  
 br = new BufferedReader(new FileReader(choosenLib));  
 while (true) {  
 String str = br.readLine();  
 if (str == null) break;  
 wordList.add(str); //添加到记录中  
 wordNum.add(0); //设置对应的初始值  
 sensWord.append(str + "\n"); //添加到界面中  
 }  
 br.close(); //关闭文件流  
 } catch (FileNotFoundException e1) {  
 // *TODO Auto-generated catch block* JOptionPane.*showMessageDialog*(null, "文件不存在");  
 e1.printStackTrace();  
 } catch (IOException e1) {  
 // *TODO Auto-generated catch block* JOptionPane.*showMessageDialog*(null, "文件读取失败");  
 e1.printStackTrace();  
 }  
 }  
  
 //高亮显示  
 public void showSensword() {  
  
 Highlighter hg = textArea.getHighlighter(); //设置文本框的高亮显示  
 hg.removeAllHighlights(); //清除之前的高亮显示记录  
 String text = textArea.getText(); //得到文本框的文本  
 DefaultHighlightPainter painter = new DefaultHighlightPainter(Color.*YELLOW*); //设置高亮显示颜色为黄色  
 for (String str : wordList) { //匹配其中的每一个敏感词  
 int index = 0;  
 while ((index = text.indexOf(str, index)) >= 0) {  
 try {  
 hg.addHighlight(index, index + str.length(), painter); //高亮显示匹配到的词语  
 index += str.length(); //更新匹配条件继续匹配  
 } catch (BadLocationException e) {  
 // *TODO Auto-generated catch block* e.printStackTrace();  
 }  
 }  
 }  
 }  
  
 //爬取网址库中的网址  
 public void spiderAll() {  
 if (wordNum.size() <= 0) { //判断是否选择了敏感词库  
 JOptionPane.*showMessageDialog*(null, "请先选择敏感词库");  
 return;  
 }  
 JFileChooser fChooser = new JFileChooser(); //选择网库文件  
 int ok = fChooser.showOpenDialog(this);  
 if (ok != JFileChooser.*APPROVE\_OPTION*) return;  
 File file = fChooser.getSelectedFile();  
 new SpiderAll(this, file).start(); //开启线程爬取  
 }  
  
  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 // *TODO Auto-generated method stub* JButton j = (JButton) e.getSource(); //判断操作来源  
 if (j == goSpider) { //爬取单个网址  
 String website = siteField.getText();  
 new SpiderOne(this, website).start();  
 } else if (j == openLib) { //打开敏感词库  
 getLib();  
 } else if (j == match) { //匹配单个网址的敏感词高亮显示  
 showSensword();  
 } else if (j == siteLib) { //爬取网址库中的全部网址  
 spiderAll();  
 }  
 }  
  
 //爬取单个网址线程  
 class SpiderOne extends Thread {  
 private String website = null; //网页链接  
  
 //构造函数初始化  
 public SpiderOne(JFrame fa, String s) {  
 website = s;  
 }  
  
 public void run() {  
 if (website.length() <= 0) { //判断网址是否正常  
 JOptionPane.*showMessageDialog*(null, "网址不能为空");  
 return;  
 }  
 htmlArea.setText(""); //清除文本  
 textArea.setText("");  
  
 String html = getHtml(website); //开始爬取  
 if (html.length() > 0) { //若爬取正常  
 JOptionPane.*showMessageDialog*(null, "爬取完毕"); //提示完成  
 htmlArea.append(html); //显示html源代码  
 String text = getText(html); //匹配网页文本  
 textArea.append(text); //显示网页文本  
 }  
 }  
 }  
  
 //爬取网址库  
 class SpiderAll extends Thread {  
 private File file = null; //网址库文本文件  
  
 //构造函数初始化  
 public SpiderAll(JFrame fa, File f) {  
 file = f;  
  
 }  
  
 public void run() {  
 try {  
 //读取网址库中的网址  
 BufferedReader brr = new BufferedReader(new FileReader(file));  
 //将匹配数据写入文本中  
 PrintStream ps = new PrintStream(new File("data.txt"));  
 ps.println("敏感词记录如下:");  
 int size = wordList.size();  
 while (true) {  
 String website = brr.readLine();  
 if (website == null) break;  
 ps.println(website + "数据如下: ");  
 String html = getHtml(website); //获取html代码  
 String text = getText(html); //匹配网页文本  
 for (int i = 0; i < size; i++) { //在网页文本中进行匹配  
 String word = wordList.get(i);  
 int index = 0, account = 0, len = word.length();  
 while ((index = text.indexOf(word, index)) >= 0) {  
 account++;  
 int temp = wordNum.get(i); //更新数据  
 wordNum.set(i, ++temp);  
 index += len; //更新匹配条件  
 }  
 ps.println(word + " 出现 " + account + "次"); //写入当前数据  
 }  
 ps.println();  
 }  
 brr.close(); //关闭文件流  
 System.*out*.println("爬取完毕");  
 ps.println("总数据如下: "); //写入总数据  
 for (int i = 0; i < size; i++) {  
 ps.println(wordList.get(i) + " 出现 " + wordNum.get(i) + "次");  
 }  
 ps.close(); //关闭文件流  
 JOptionPane.*showMessageDialog*(null, "爬取完毕！请打开文件查看!");  
 } catch (Exception e) {  
 // *TODO: handle exception* JOptionPane.*showMessageDialog*(null, "爬取失败");  
 }  
 }  
 }  
}

1. Main.java

import java.io.IOException;  
  
public class Main {  
  
 public static void main(String[] args) throws IOException {  
 new Spider();  
 }  
}

1. KeywordAnalysis.java

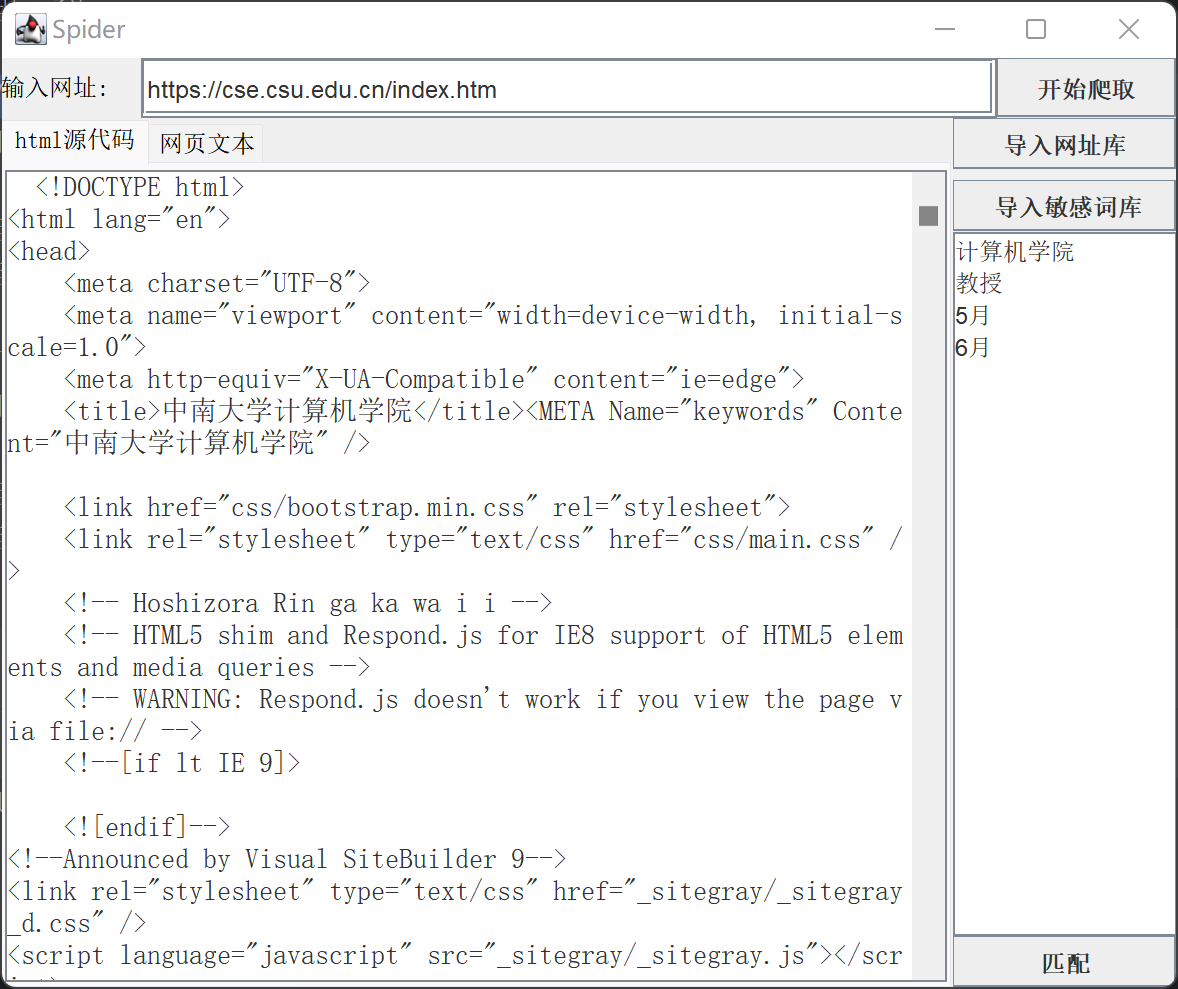
import java.io.BufferedReader;  
import java.io.BufferedWriter;  
import java.io.FileOutputStream;  
import java.io.FileReader;  
import java.io.IOException;  
import java.io.OutputStreamWriter;  
import java.util.HashMap;  
import java.util.Map;  
  
public class  
KeywordAnalysis{  
  
 public static void main(String[] args) {  
 // 读取文本文件  
 String filePath = "D:\\code\\url\\data.txt";  
  
 try (BufferedReader reader = new BufferedReader(new FileReader(filePath))) {  
 String line;  
 Map<String, Integer> keywordCountMap = new HashMap<>();  
  
 // 提取关键词并统计出现次数  
 while ((line = reader.readLine()) != null) {  
 // 判断是否为关键词记录行（以关键词开头）  
 if (line.startsWith("计算机学院") || line.startsWith("教授") ||  
 line.startsWith("5月") || line.startsWith("6月")) {  
 String[] parts = line.split("\\s+"); // 拆分关键词和出现次数  
 String keyword = parts[0];  
 String countString = parts[2].replace("次", ""); // 去除次字符串  
 int count = Integer.*parseInt*(countString);  
  
 keywordCountMap.put(keyword, count);  
 }  
 }  
  
 // 保存数据到 CSV 文件  
 String csvFilePath = "D:\\code\\url\\data.csv";  
 try (BufferedWriter writer = new BufferedWriter(new OutputStreamWriter(new FileOutputStream(csvFilePath), "UTF-8"))) {  
 // 写入表头  
 writer.write("关键词,出现次数");  
 writer.newLine();  
  
 // 写入数据行  
 for (String keyword : keywordCountMap.keySet()) {  
 int count = keywordCountMap.get(keyword);  
 writer.write(keyword + "," + count);  
 writer.newLine();  
 }  
  
 System.*out*.println("数据已保存到 " + csvFilePath);  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
}

1. MySQLImportCSV.java

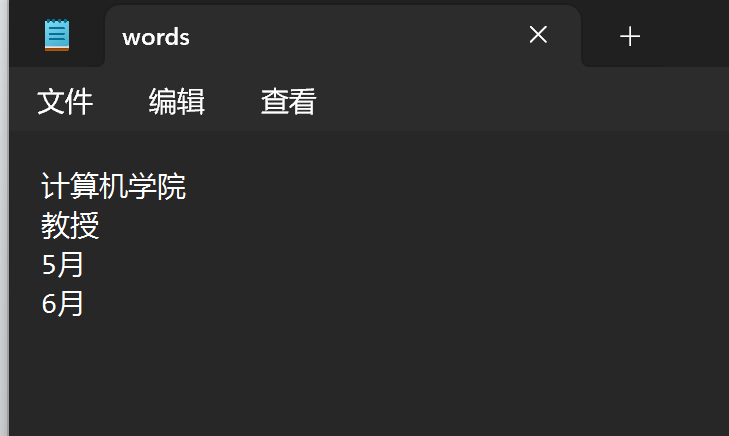
import java.io.BufferedReader;  
import java.io.FileReader;  
import java.io.IOException;  
import java.sql.Connection;  
import java.sql.DriverManager;  
import java.sql.PreparedStatement;  
import java.sql.SQLException;  
  
public class MySQLImportCSV {  
 public static void main(String[] args) {  
 // MySQL数据库连接信息  
 String url = "jdbc:mysql://localhost:3306/url";  
 String username = "root";  
 String password = "123456";  
  
 // CSV文件路径  
 String csvFilePath = "D:\\code\\url\\data.csv";  
  
 try (Connection connection = DriverManager.*getConnection*(url, username, password);  
 BufferedReader reader = new BufferedReader(new FileReader(csvFilePath))) {  
  
 String line;  
 String sql = "INSERT INTO a (keyword, count) VALUES (?, ?)";  
 PreparedStatement statement = connection.prepareStatement(sql);  
  
 // 跳过CSV文件的表头行  
 reader.readLine();  
  
 // 逐行读取CSV文件并插入数据库  
 while ((line = reader.readLine()) != null) {  
 String[] data = line.split(",");  
 String keyword = data[0];  
 int count = Integer.*parseInt*(data[1]);  
  
 statement.setString(1, keyword);  
 statement.setInt(2, count);  
 statement.executeUpdate();  
 }  
  
 System.*out*.println("数据导入完成。");  
  
 } catch (IOException | SQLException e) {  
 e.printStackTrace();  
 }  
 }  
}

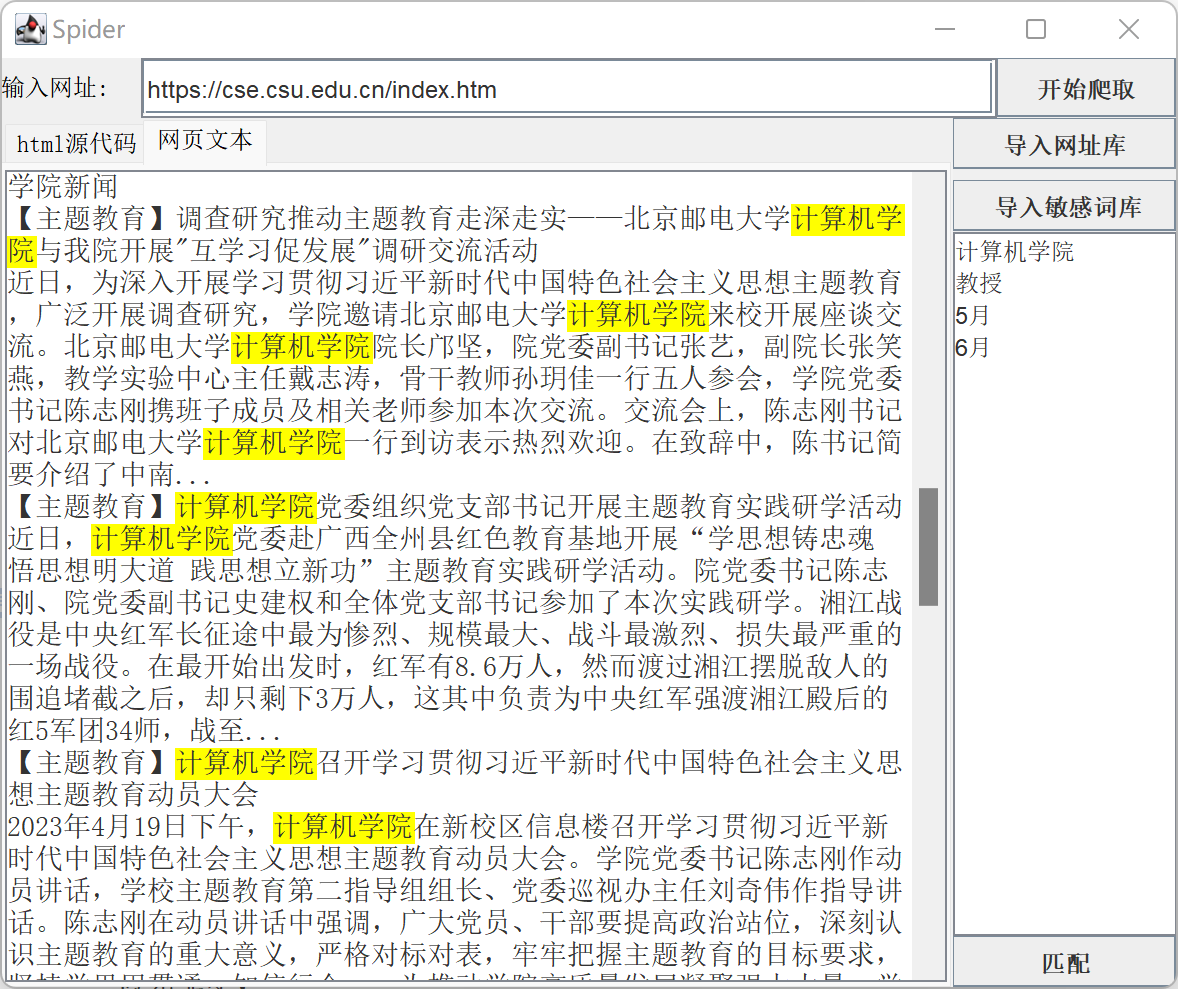
**第九章用JavaURL编程爬取并分析网页敏感词运行结果与测试分析**

输入一个网址，能够爬取该网址上所有的HTML源代码并对网址中的文本进行提取：



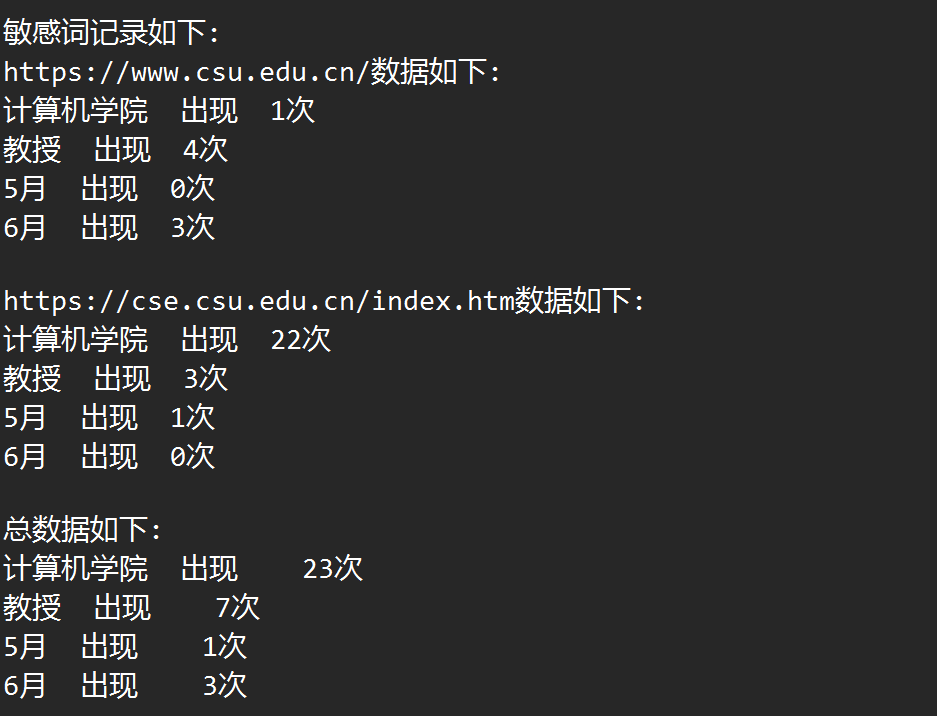
建立敏感词库，用文本文件保存。将该网址所对应的文本中的敏感词提取并高亮显示：

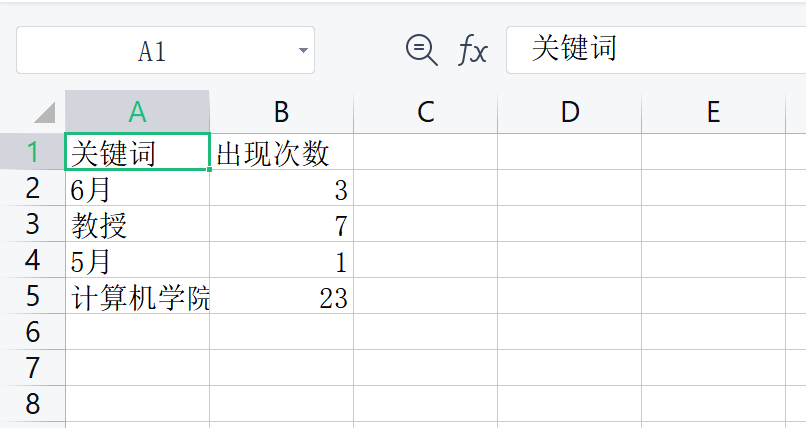




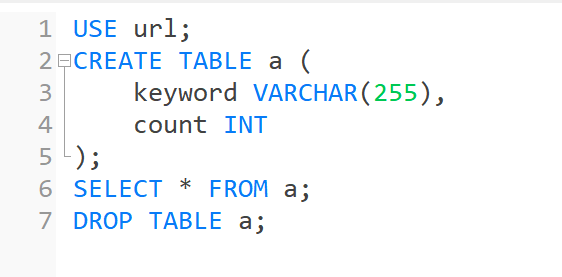
**增加功能：**

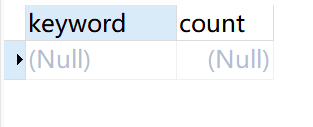
程序可爬取url.txt文件中网址中的文本内容，将敏感词记录存入另一个csv文件:





将csv文件中的数据与本地MySQL数据库连接并实现查询：







**第十章结论与心得**

1. **Java Socket编程开发聊天室：**

通过这个实验，我深入了解了Socket编程的概念和原理，以及如何在Java中使用Socket类进行网络通信。通过实现聊天室服务器端和客户端，我学会了如何建立连接、发送和接收消息，并实现了群聊和私人聊天功能。这个实验对我理解多线程编程和并发控制也起到了很好的锻炼作用，加深了我对Java网络编程的理解。

1. **Java URL编程爬取并分析网页敏感词：**

通过这个实验，我学会了如何使用Java的URL类来访问网页，并获取网页的HTML源代码。在实验中，我了解了如何解析HTML源代码，通过正则表达式提取其中的文本内容。让我能够对获取的网页文本进行敏感词检测和过滤，让我更好地理解了文本处理和字符串操作的方法。

**心得体会：**

这两个实验使我对Java Socket编程和Java URL编程有了更深入的了解，并且提升了我的网络编程和文本处理能力。通过实际的实验，我感受到了这些技术的实际应用和潜力，对我今后的编程发展有着积极的影响。