

**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 | \*\*\*\*\*\*\*\* |

第一章

开发环境：Intelij IDEA 和JDK8

第二章

网络聊天程序业务分析

用JavaSocket编程开发聊天室

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。

说明：1. 客户端和服务端都用swing做成图形界面

2． 客户端与服务端都可以用Socket进行通信

3. 客户端和服务端都实现Runnable接口，开启线程。

4. 通过流将数据封装，通过用户名对不同用户进行区别

第三章

网络聊天程序系统设计

1. 系统功能定义

服务端的代码在Server.java文件中

public void run() 用来对用户的登录进行监听，如果用户上线，服务器端就会有用户登录信息。

public void actionPerformed(ActionEvent e) 有两个JButton会用到这个方法，一个功能是对客户端发送消息，另一个是踢人

public void handleAll() 功能是群发

public void handleExpel() 功能是踢人

public void sendMessage(String msg)功能是对所用用户进行发送msg，

Server.java中还有一个类ChatThread 功能是开启一个线程，对每个事件进行判断，看是否是踢人，群发，还是私聊，这里会用到一个switch case语句

客户端的代码在Client.ava中

public void run() 开启线程，处理消息

public void actionPerformed(ActionEvent e) 处理鼠标点击事件

public void handleSS() 在私聊窗口中发消息

public void handleSend() 群发消息

public void handleSec(String name) 建立私聊窗口

1. 关键问题设计

最关键的是Client.java中的public void run()的设计

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);  
 }  
 }  
}

第四章

聊天程序源代码清单

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;

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

public Client() throws Exception {

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;

}

});

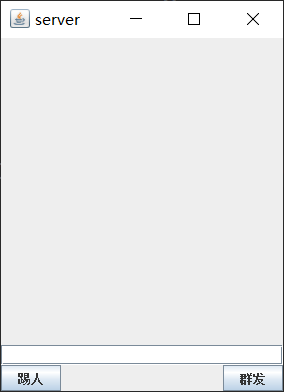
}//私聊窗口

}

第五章

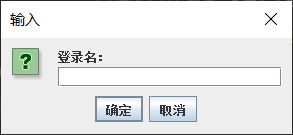
聊天程序运行结果与测试分析

服务器：

****

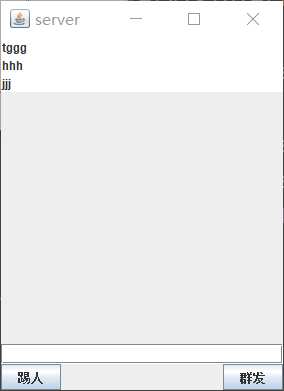
客户端：

登录：

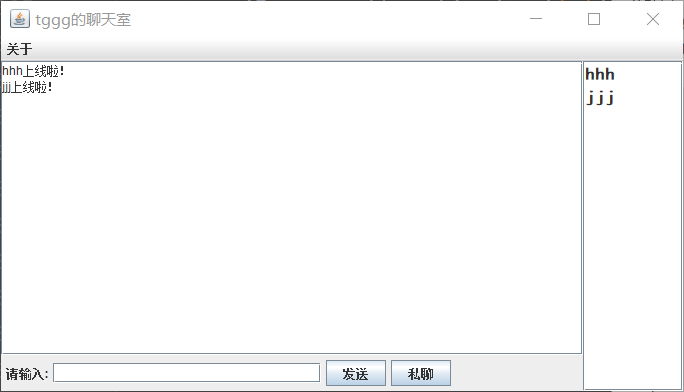
****

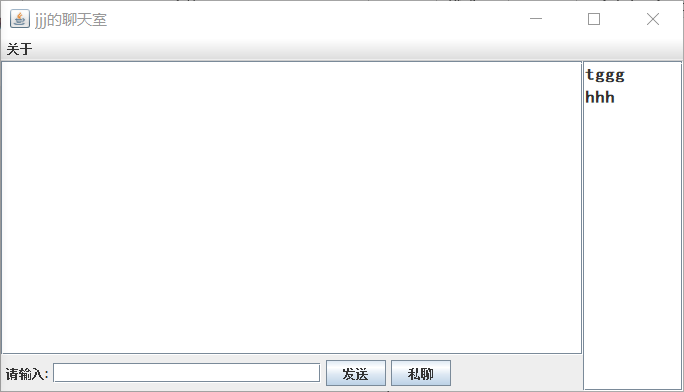
三个用户登录后：

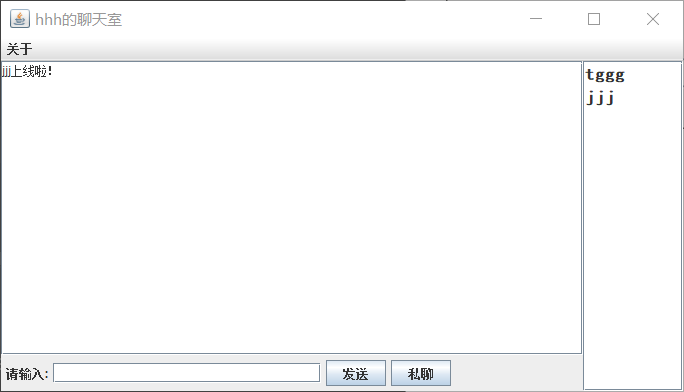
服务器会提醒三个用户登录：

****

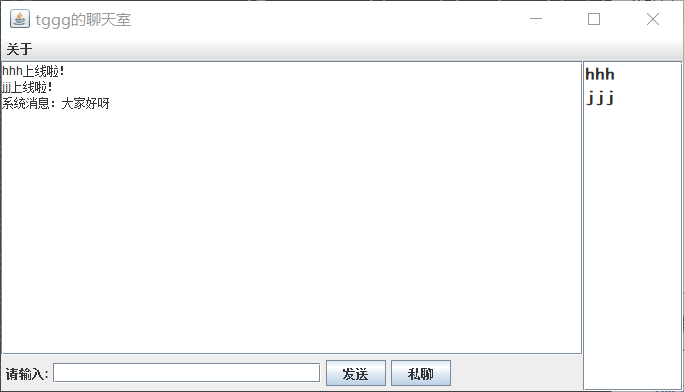
每个客户端会显示其好友：

****

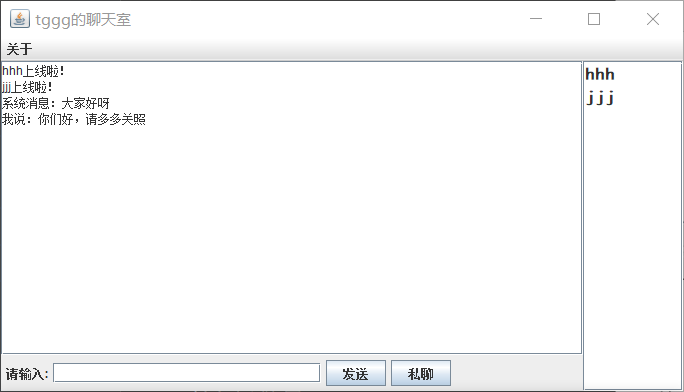
****

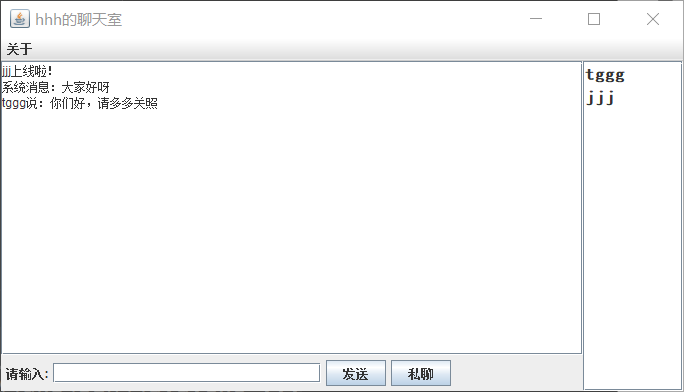
****

服务器进行群发后客户端会收到信息：

****

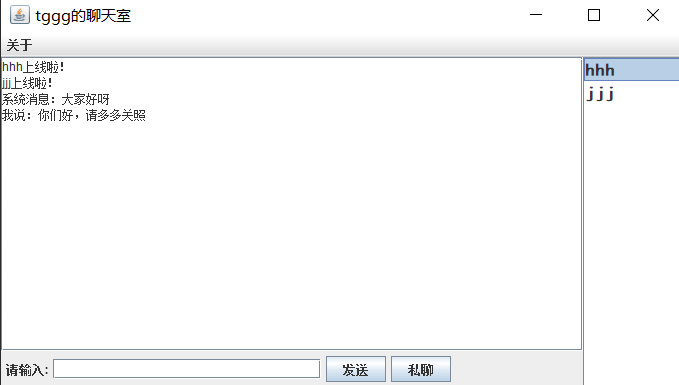
群发消息其他人都能收到消息：

****

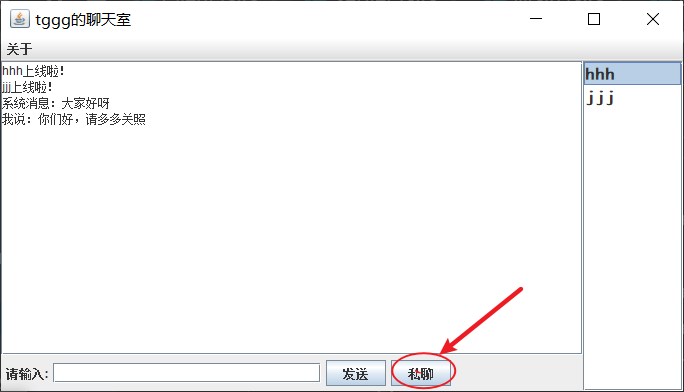
****

私聊：

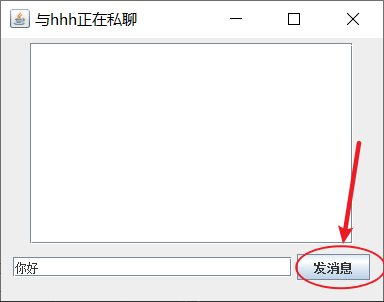
先选定好友：

****

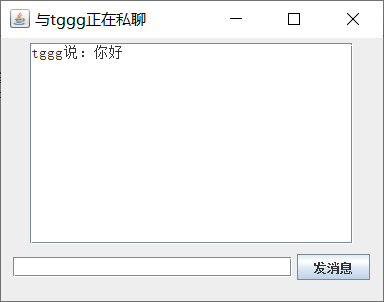
再点击私聊：

****

出现新窗口后点击发送：

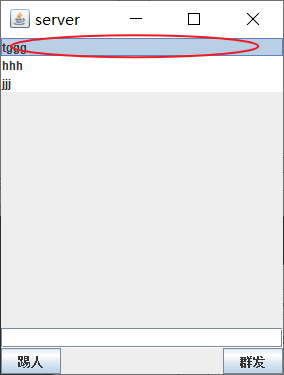
****

他的好友会收到消息：

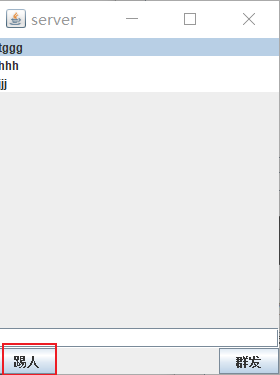
****

踢人功能：

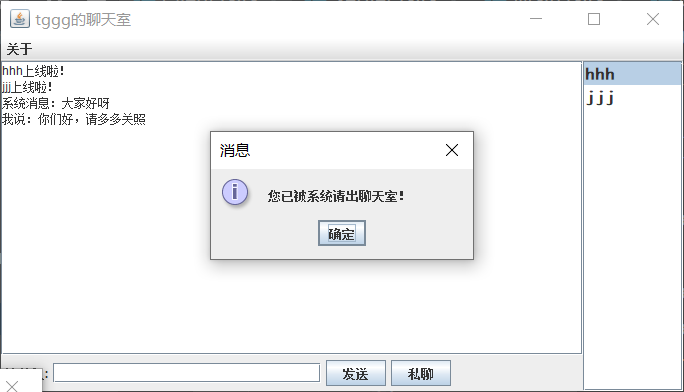
先选定人：

****

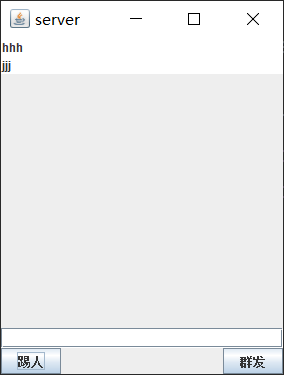
点击踢人：

****

踢人后会进行提示：

****

服务器的成员列表会减少，可以看到只剩下了两个人了

****

**第六章**

业务分析

用JavaURL编程爬取并分析网页敏感词

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**

系统设计

系统功能定义

所以代码在Spider.java中包括：

public String getHtml(String website) 使用URL爬取网页的html代码

public String getText(String str) 对html进行正则匹配,提取出其中的文本

public void getLib() 从文件中读取敏感词

public void showSensword() 高亮显示

public void spiderAll() 爬取网址库中的网址

public void actionPerformed(ActionEvent e)断操作来源

class SpiderOne extends Thread 爬取单个网址线程

class SpiderAll extends Thread爬取网址库

关键问题设计

最关键的是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  
 }  
}

第八章

源代码清单

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 JComboBox<String> charset = new JComboBox<String>();  
  
  
 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  
 private String regExHtml = "<[^>]+>"; //匹配标签  
 private String regExScript = "<script[^>]\*?>[\\s\\S]\*?<\\/script>"; //匹配script标签  
 private String regExStyle = "<style[^>]\*?>[\\s\\S]\*?<\\/style>"; //匹配style标签  
 private String regExSpace = "[\\s]{2,}"; //匹配连续空格或回车等  
 private String regExImg = "&[\\S]\*?;+"; //匹配网页上图案的乱码  
 //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());  
  
 //添加编码方式  
// charset.addItem("UTF-8");  
// charset.addItem("GBK");  
// charset.setEditable(false); //设置为不可编辑  
 //处理其事件,更新编码方式  
// charset.addActionListener(new ActionListener() {  
// //获取选择的编码方式,默认情况下为UTF-8  
// @Override  
// public void actionPerformed(ActionEvent e) {  
// // *TODO Auto-generated method stub*// textType = (String) charset.getSelectedItem();  
// }  
// });  
  
 //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*));  
 //System.out.println("开始爬取");  
 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");  
 }  
 }  
 }  
}

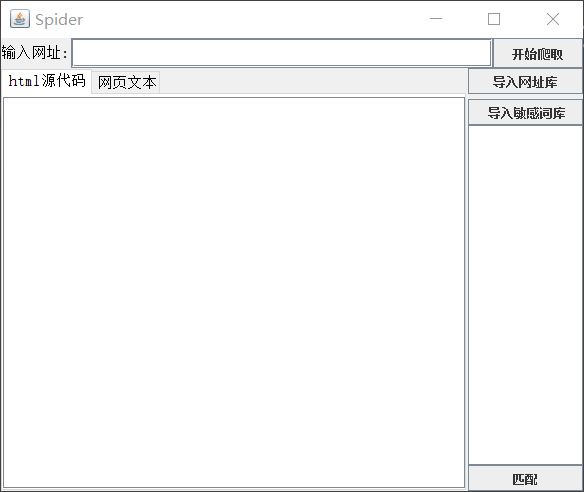
Main.java

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

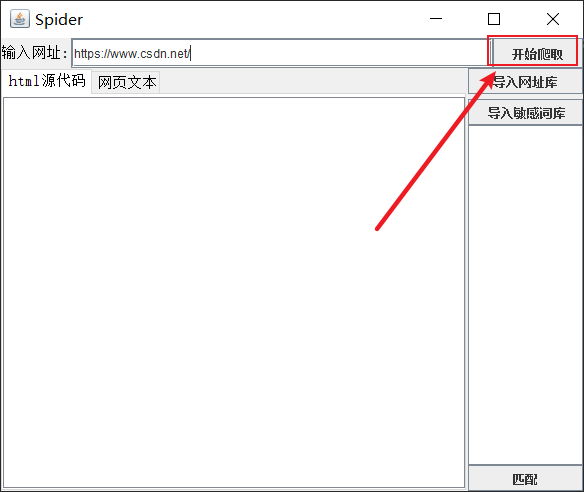
第九章

运行结果与测试分析

主界面：

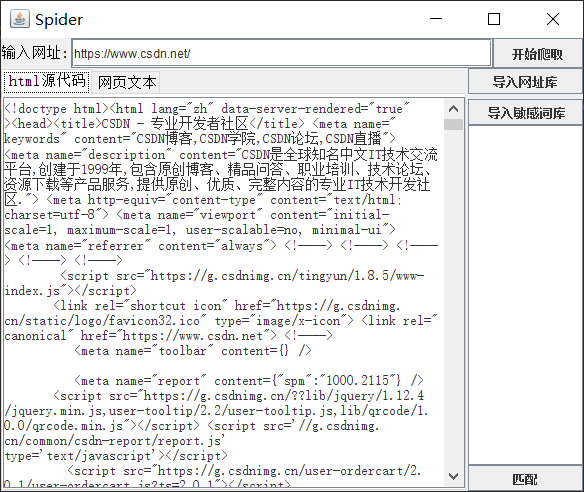
****

爬取CSDN网站：

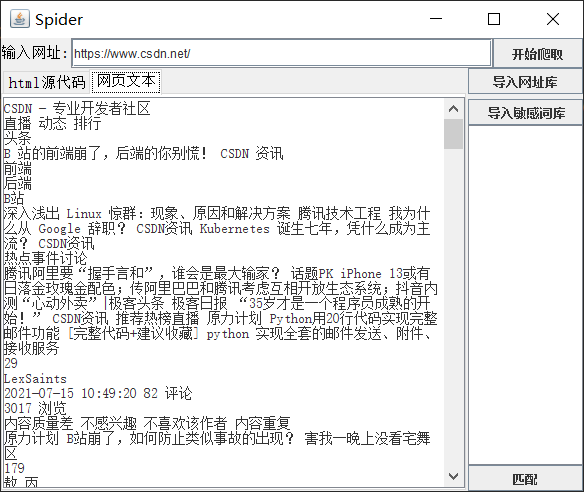
****

点击后提示爬取完毕：

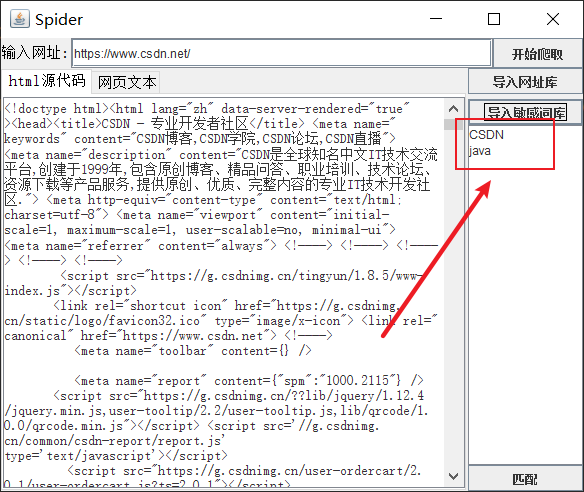
html源代码：

****

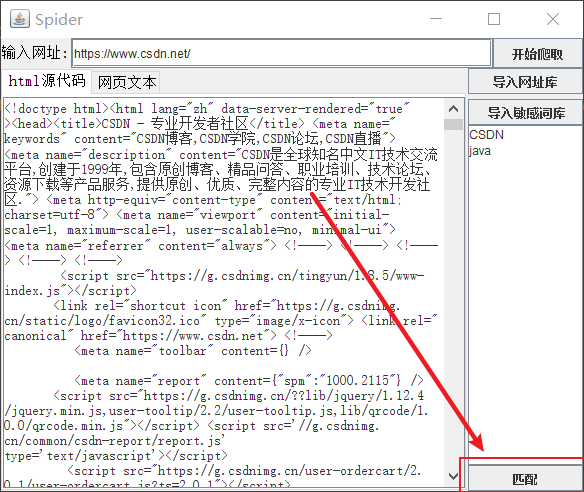
网页文本：

****

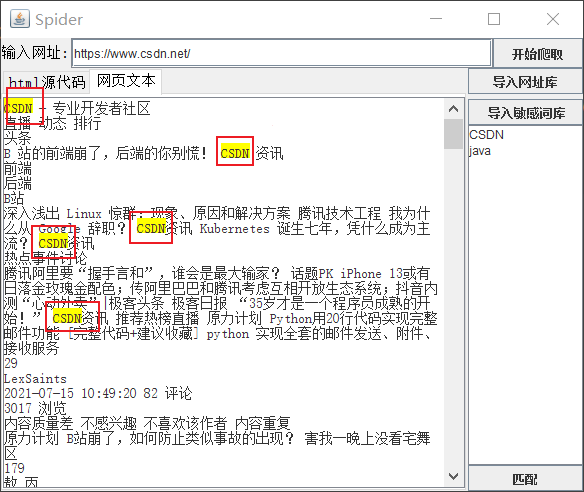
导入敏感词库：CSDN，Java

****

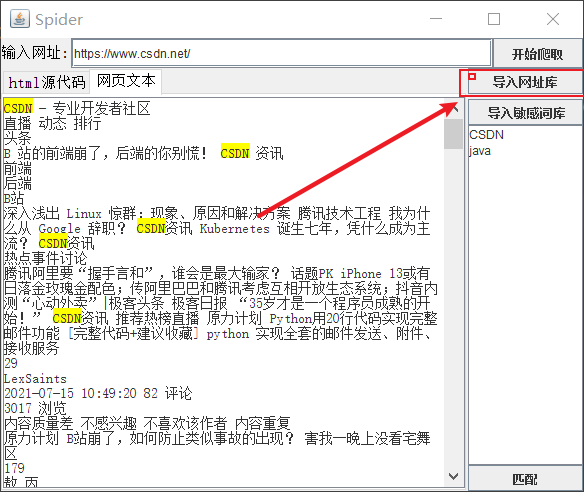
点击匹配：

****

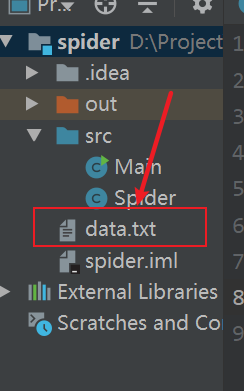
可以发现网页文本中敏感词汇变黄：

****

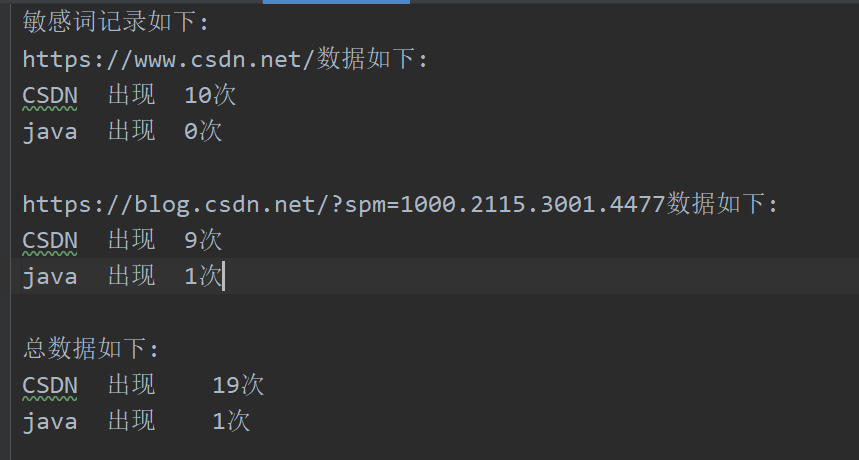
导入网址库：

****

点击匹配,会生成一个文件，data.txt保存结果

****

产生的结果：

****

**第十章**

结论与心得

对Java的规范有了更深的理解，对IDEA的使用更加熟练

对JavaAPI　和　计算机网络的理解更深入了

对Java　爬虫的使用有了一定的了解，以前写过python爬虫，现在使用Java后发现差不多，不过python提供的库更多，更方便使用

总之，收获很大也很多，但是遇到的困难也很多，但是困难也是收获！