****

**JavaSocket编程开发聊天室**

**JavaURL爬取并分析网页敏感词**

|  |  |
| --- | --- |
| 学生姓名 |  |
| 学 号 |  |
| 专业班级 |  |
| 指导教师 |  |
| 学 院 | 计算机学院 |
| 完成时间 | 2022.6.20 |

计算机学院

目录

[题目一要求 3](#_Toc107343319)

[题目一的设计思路 3](#_Toc107343320)

[设计经历 4](#_Toc107343321)

[服务器： 5](#_Toc107343322)

[客户端 6](#_Toc107343323)

[题目一的实现过程 7](#_Toc107343324)

[服务器 7](#_Toc107343325)

[数据库处理类 7](#_Toc107343326)

[服务器变量 8](#_Toc107343327)

[处理请求 8](#_Toc107343328)

[编写控制窗口controller 9](#_Toc107343329)

[客户端 10](#_Toc107343330)

[客户端变量 10](#_Toc107343331)

[客户端处理服务器返回json包 12](#_Toc107343332)

[登录注册功能 12](#_Toc107343333)

[历史信息功能 13](#_Toc107343334)

[发送表情功能 14](#_Toc107343335)

[题目一的难点及解决 17](#_Toc107343336)

[用户私聊信息的缓存（免打扰模式） 17](#_Toc107343337)

[登录/注册/注销时在线成员列表的动态刷新 19](#_Toc107343338)

[窗口放大缩小的动态变化 20](#_Toc107343339)

[聊天窗口的显示 21](#_Toc107343340)

[私聊功能接收结果报文死机问题 21](#_Toc107343341)

[题目一的运行及测试结果 22](#_Toc107343342)

[私聊功能 22](#_Toc107343343)

[聊天室群聊功能 23](#_Toc107343344)

[历史消息查询 24](#_Toc107343345)

[服务器端 25](#_Toc107343346)

[题目二要求 26](#_Toc107343347)

[题目二的设计思路 26](#_Toc107343348)

[题目二的实现过程 26](#_Toc107343349)

[爬取html 26](#_Toc107343350)

[提取文本 27](#_Toc107343351)

[界面设计和高亮显示 28](#_Toc107343352)

[爬取多个网站 30](#_Toc107343353)

[题目二的难点及解决 31](#_Toc107343354)

[给敏感词的高亮换色 31](#_Toc107343355)

[题目二的运行及测试结果 34](#_Toc107343356)

[单个网页爬取效果图 34](#_Toc107343357)

[爬取多个网站 37](#_Toc107343358)

[总体心得体会 39](#_Toc107343359)

# 题目一要求

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。

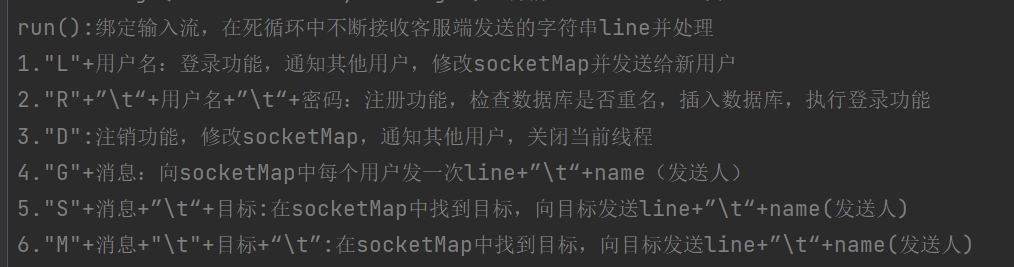
# 题目一的设计思路

项目结构：

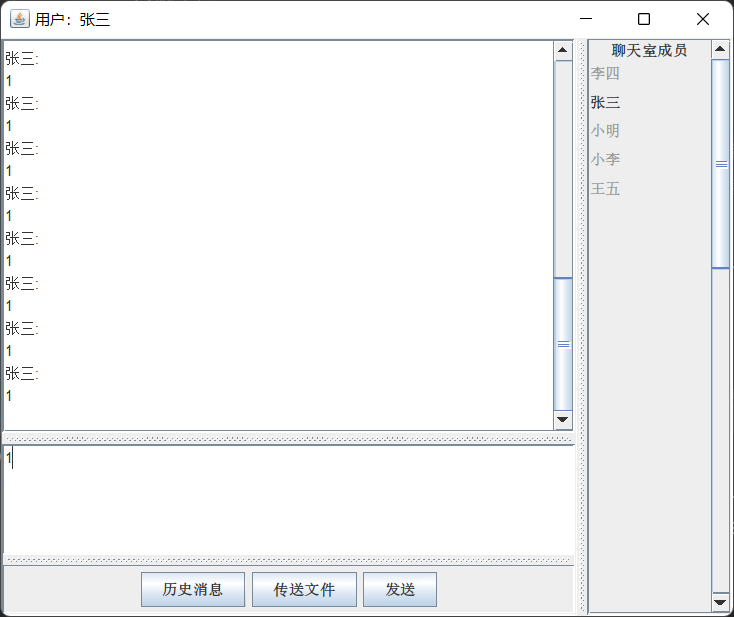
使用maven管理项目，db下是数据库操作的辅助类，Frame下是窗口类，最主要的是聊天室窗口，另外两个是登录窗口和注册窗口。Client是客户端，Server是服务器，controller是服务器控制端，用来完成群发系统消息和让某些用户下线的功能。Resources中的icon存一些图标，qqdefaultface是使用到的105个QQ表情，config.properties存放数据库连接的一些参数

## 设计经历

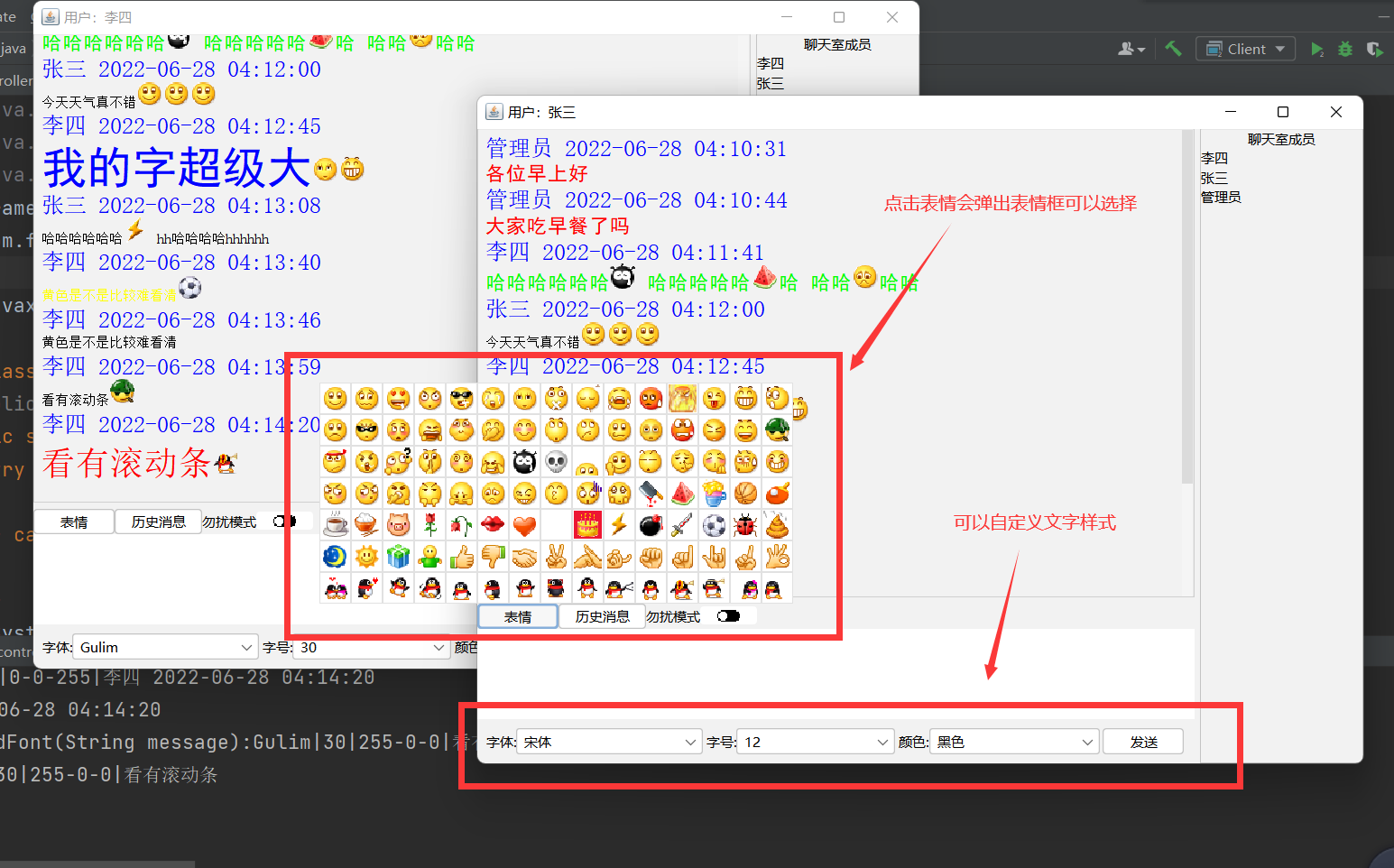
整个聊天室总共改过三大版，第一版客户端服务器之间的通信是通过传输由信息和分隔符组成的字符串来完成的；第二版把通信部分全改成了json格式，操作方便也更合规范，不用担心客户输入了分隔符且更容易扩充；第三版为了能够支持发送表情功能，将JTextArea（仅支持显示文字）都换成了JTextPane（支持显示标签，方便信息插入）。



1第一版字符串传输



2第二版的界面



3第三版界面



4通信时的传输格式

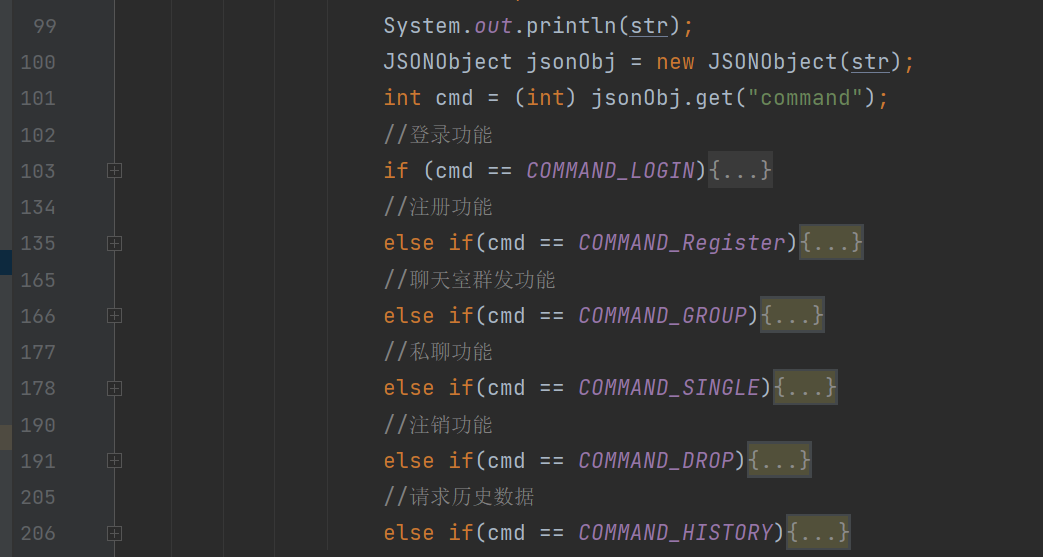
## 服务器：

服务器主程序在一个while(true)中持续监听端口，每接收到一个新客户连接请求就为该客户新开一个线程listener，负责和该客户进行通信。Listener线程持续接收json数据包，用jsonObj.get("command")取出命令，判断后执行登录、注册、群发、私聊、注销、请求历史信息等六个操作。

服务器的显示面板作为另开的一个类controller，修改ChattingRoomFrame增加两个功能即可。

此外，还需要新建辅助数据库访问的UserDAO类封装对数据库的增删改查。

服务器编写的重点在于接收不同command的请求然后访问数据库请求数据并返回结果，对各种数据进行转发，对用户进行控制管理等。



5服务器的分支处理

## 客户端

客户端启动弹出登录窗口，支持账户管理会校对账号密码，还可以注册账户。登陆成功后用服务器返回的数据如在线人员等信息初始化聊天室界面，同时启动新线程持续监听服务器发回的json包并据此更新界面，如新用户登录注册刷新在线人员表，收到群发消息显示在聊天板上，收到私聊消息根据勿扰模式开启新窗口/进行消息缓存，收到下线命令关闭socket，收到历史消息显示在HistoryBoard上。

客户端最难的地方在于处理支持发送表情，此外主要动态刷新在线人员列表，对私聊消息缓存等，这些在后续详细展开，其他的都比较常规略去不谈了。



6客户端的分支处理

# 题目一的实现过程

## 服务器

### 数据库处理类

比较常规，基本的连接然后用sql语句进行查询。这里为了方便数据库的切换，改变数据库信息如名字，密码，驱动等额外编写了一个config.properties文件存储这些信息

1. static {
2. *// 从配置文件中获得属性文件输入流*
3. InputStream input = DBHelper.class.getClassLoader()
4. .getResourceAsStream("config.properties");
5. try {
6. info.load(input);
7. url = info.getProperty("url");
8. String driverClassName = info.getProperty("driver");
9. Class.forName(driverClassName);
10. System.out.println("驱动程序加载成功...");
11. } catch (ClassNotFoundException e) {
12. System.out.println("驱动程序加载失败...");
13. } catch (IOException e) {
14. System.out.println("加载属性文件失败...");
15. }

数据库查询操作也很常规，数据库中的表有user,history,friend（之前存储好友关系，后废弃不用）主要封装接口如下：

findAllUser查询所有用户

findById根据用户名查询用户信息

insertUser向user表中插入新用户

insertMess向history表中插入新的历史消息

queryHistory根据日期在history表中查询历史消息

代码较多不再赘述，JAVA访问数据库的展示基本格式如下：

1. String sql = "xxxxxxxxx"；*//编写sql语句*
2. conn = DBHelper.getConnection();*//连接数据库*
3. pstmt = conn.prepareStatement(sql);*//装入sql语句*
4. pstmt.setString(1, user\_name);*//填写sql语句中的参数*
5. ResultSet rs = pstmt.executeQuery();*//执行sql语句返回结果*
6. *//todo 对rs进行一些处理得到结果result*
7. return result

### 服务器变量

服务器存储变量如下：

UserDAO dao:数据库查询辅助类

HashSet<String> userList:用户名列表

Map<String,Socket> socketMap:所有在线用户的socket表

int port:端口号

ExecutorService executorService:线程池

ServerSocket serverSocket:服务器Socket

### 处理请求

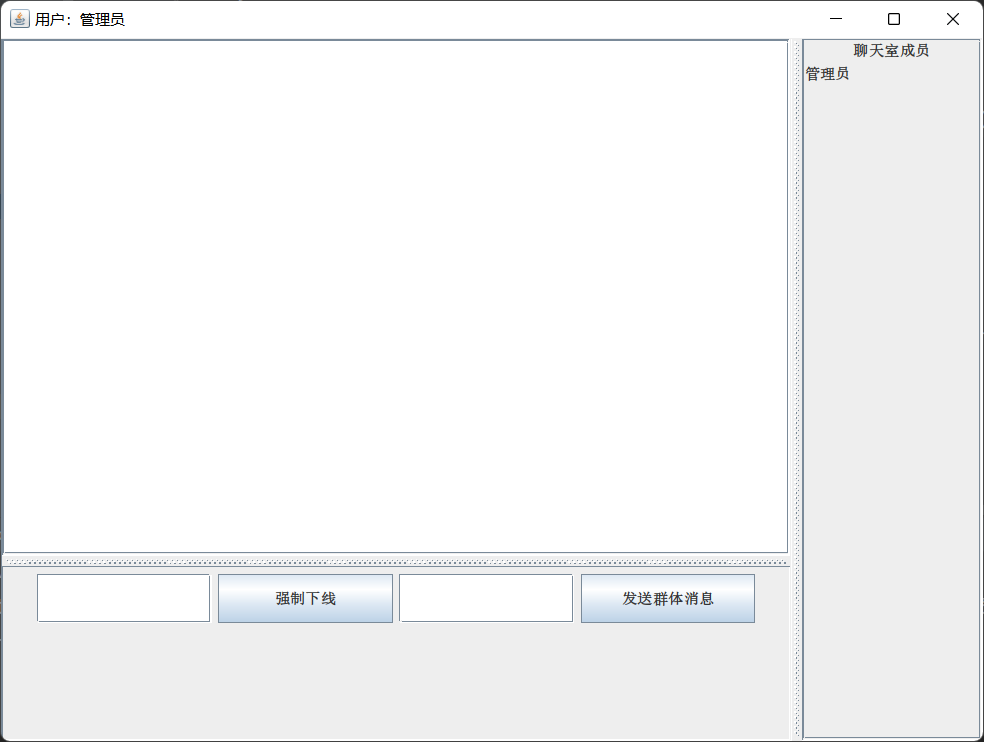
简单封装一个sendMessage(Socket client, String s)负责向client发送信息s。服务器的listner总共处理六种请求：

1. 群发功能：将该json包添加发送方名字name后根据socketMap广播即可。
2. 私聊功能：根据json包中的target字段在socketMap中查询目标用户的socket并转发该json包
3. 注销功能：socketMap中移除该用户的socket，根据socketMap广播该用户的下线，关闭该socket连接
4. 查询历史记录：根据json包中的date字段查询数据库取出当天的所有消息，封装成json包返回
5. 登录/注册功能较复杂如下详细展开，其中注册功能包括了登录，仅以注册功能展开：
6. 从json中取出姓名和密码字段，通过dao插入新用户数据，更新用户总表userList ，广播新的用户总成员表，至此成功注册新用户。然后根据socketMap广播新用户登录，再将新用户添加进socketMap，然后将聊天室总成员名单和在线人员名单发送给新用户。这里处理的顺序不能调换，否则会出问题。（最开始编写时每个客户端会收到总成员名单和在线成员名单，显示总名单用黑色和灰色区分在线与否。后来修改为仅显示在线成员了，由于不影响功能所以未修改服务器部分代码）
7. else if(cmd == COMMAND\_Register){
8. name = (String) jsonObj.get("user\_name");
9. String pwd=(String) jsonObj.get("pwd");
10. dao.insertUser(name,pwd);*//插入新用户数据*
11. userList = dao.findAllUser();*// 更新总用户名单信息*
12. *//广播更新后的总成员表*
13. JSONObject broadcast = new JSONObject();
14. broadcast.put("command",COMMAND\_Register);
15. broadcast.put("userList",userList);
16. for(Socket socket:socketMap.values()){
17. sendMessage(socket,broadcast.toString());
18. }
19. *//广播当前用户上线了*
20. broadcast = new JSONObject();
21. broadcast.put("command",COMMAND\_LOGIN);
22. broadcast.put("name",name);
23. for(Socket socket:socketMap.values()){
24. sendMessage(socket,broadcast.toString());
25. }
26. *//进行登录*
27. Map<String, String> user = dao.findById(name);
28. socketMap.put(name,this.socket);*//socket表加入新用户*
29. JSONObject sendJsonObj = new JSONObject(user);*//创建结果json包*
30. sendJsonObj.put("result", 1);*// 添加result:0键值对，1表示成功，0表示失败*
31. Set<String> online = socketMap.keySet();*// 在线人员列表*
32. sendJsonObj.put("online", online);
33. sendJsonObj.put("userList", userList);*// 聊天室成员总名单*
34. sendMessage(socket,sendJsonObj.toString());*// 创建DatagramPacket对象，用于向客户端发送数据*
35. }

### 编写控制窗口controller

方便起见controller直接用管理员账号登录并实例化一个ChattingRoomFrame control，然后用remove移除control下方输入区部分，并添加下线用户，群发消息的输入框和按钮。由于现实生活中的服务器程序是不会有可视化界面的，这里的管理员相当于一个客户端，其对服务器的控制也是通过socket通信，能额外发送COMMAND\_CONTROLD强制注销和COMMAND\_CONTROLS群发消息两个命令。服务器接收到后对于强制注销，取出socketMap中对应用户的socket发送注销命令然后关闭；对于群发消息，按正常群发消息流程走即可。

代码比较简单略去，演示操作在后面的测试结果中展示，控制窗口界面展示如下：



## 客户端

### 客户端变量

客户端变量众多，展示如下，都有详尽的注释



1. *//运行所需变量*
2. int h=600;*//窗口高*
3. int w=800;*//窗口宽*
4. String username;*//当前用户的名字*
5. Socket socket;*//当前用户连接的socket*
6. SimpleDateFormat sf = new SimpleDateFormat("yyyy-MM-dd hh:mm:ss");*//日期格式*
7. FontAndText dateFont = new FontAndText("", "宋体", 20, Color.BLUE);*//固定用户名+日期的信息头样式*
8. int pos1;*//全局存储当前插入的起始位置，插入表情时以此为基准*
9. FontAndText myFont = null;*//全局存储用户选择的文本样式*
10. *//运行所需容器*
11. List<String> userlist;*//用户总成员列表（修改后不再用了，可以删掉了）*
12. Set<String> online;*//在线人员名单*
13. List<JLabel> userLabelList;*//用户标签列表*
14. Map<String,SingleChatFrame>singlewindow=new HashMap<>();*//管理私聊窗口（私聊对象：私聊窗口）*
15. Map<String,ArrayList<String>>messBuffer=new HashMap<>();*//消息缓冲（私聊对象：发送信息缓冲）*
16. List<PicInfo> myPicInfo = new LinkedList<>();*//自己的表情信息*
17. List<PicInfo> receivePicInfo = new LinkedList<>();*//接收到的表情信息*
18. *//标志*
19. boolean flag=true;*//在线*
20. *//重要控件*
21. Box menberListBox=createVerticalBox();*//总成员列表面板*
22. HistoryBoard hisbd=new HistoryBoard();*//历史记录板*
23. PicsJWindow picWindow=new PicsJWindow(this);*//表情框*
24. *//聊天区*
25. JScrollPane chatScroll = new JScrollPane();*//聊天板的滚动底板*
26. JTextPane chatBoard=new JTextPane();*//聊天板*
27. StyledDocument docChat = chatBoard.getStyledDocument();*//聊天板的通用样式化文档*
28. *//输入区*
29. JPanel inputArea = new JPanel(new BorderLayout());*//输入区域*
30. Box btnZone = Box.createHorizontalBox();*//表情，历史记录，勿扰模式等开关控制区*
31. JTextPane content = new JTextPane();*//用户输入框*
32. StyledDocument docContent = content.getStyledDocument();*//输入框的通用样式化文档*

### 客户端处理服务器返回json包

客户端持续监听服务器返回json包，如果是接收到他人登录/注册/下线的消息，需要refreshFriendList刷新在线人员面板（刷新实现在难点解决中展开）,如果是群发消息则将信息展示在聊天板中（在下面的发送表情功能中展开），如果是私聊消息则弹出私聊窗口开始私聊（在难点解决中展开），如果是强制下线则直接关闭socket并弹出提示提醒用户被强制下线。如果是查询历史记录则显示历史信息板（在下面历史信息中展开）。

### 登录注册功能

根据控件的getText获取输入框的用户名和密码信息，进行判空处理后调用login函数，根据信息封装json包并传送给服务器，然后等待服务器返回的结果包。上层处理接收login返回的map，如果为空说明登录失败弹出提示，否则用服务器返回的json包去初始化聊天室窗口ChattingRoomFrame。其中主要的是login函数，代码如下：

1. public static Map login(String name, String pwd) {
2. try {
3. JSONObject jsonObj = new JSONObject();
4. jsonObj.put("command", COMMAND\_LOGIN);
5. jsonObj.put("user\_name", name);
6. jsonObj.put("user\_pwd", pwd);
7. *// 以字符串传输json*
8. OutputStream out = socket.getOutputStream();
9. PrintStream printStream = new PrintStream(out);
10. printStream.println(jsonObj.toString());
11. printStream.flush();
12. *// 接收返回的json结果*
13. InputStream in = socket.getInputStream();
14. Scanner sc = new Scanner(in);
15. String msg="";
16. if(sc.hasNext()){
17. msg = sc.nextLine();
18. }
19. JSONObject receivedjsonObj = new JSONObject(msg);
20. System.out.println(receivedjsonObj);
21. if ((Integer) receivedjsonObj.get("result") == 1) {
22. Map user = receivedjsonObj.toMap();
23. return user;
24. }
25. } catch (IOException e) {
26. e.printStackTrace();
27. }
28. return null;
29. }

### 历史信息功能

定义历史消息面板类HistoryBoard，通过addHistory接口向面板中添加历史信息。

从json包中取出history字段信息转化为List<String> historytext，并调用addHistory添加历史信息，代码略，展示效果如下：



### 发送表情功能

#### 辅助类

ChatPic：

继承自ImageIcon，封装图标，主要是加一个可修改的代号。新增属性图片代号int im。

PicsJWindow：

继承自JWindow（单纯的窗体没有关闭缩小等按键）制作自定义表情框。用GridLayout(7, 15)制作7\*15的网格存放105个表情，用JLabel[] ico = new JLabel[105]的JLabel数组存放105个表情。

Init()初始化函数根据文件路径名找到表情存放的文件位置，读取qqdefaultface中的每个gif文件制成JLabel依次存入ico中，为每个表情添加鼠标事件移入时蓝边框移出时还原，点击后调用父窗口的insertSendPic将所选表情添加进输入框。

setVisible用于设置窗口可见/隐藏

FontAndText：

字体和文本类，主要封装字体样式包括文本，字体类型，字号大小，字体颜色并将其封装成SimpleAttributeSet attrSet以方便JTextPane使用。

PicInfo：

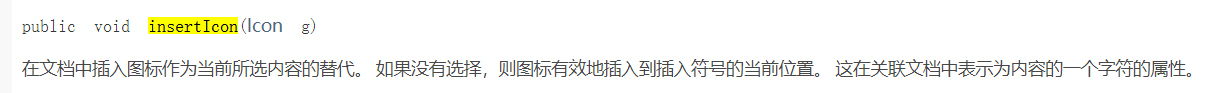
图片信息。由于双方都有表情库，所以传输的时候不是发送图片，而是传文本串，表情在文本串中的位置，表情序号。然后接收方先插入文本串，然后根据序号找到表情，插入文本串中的对应位置。所以封装图片信息类存表情的位置pos和序号val。

#### 发送方发送表情时候的处理

##### 发送区表情的显示

点击表情按键时调用setVisible设置表情框PicsJWindow可见，然后用户选择表情触发点击事件，表情框调用聊天板的insertSendPic在输入区中插入图片。调用JTextPane的insertIcon函数可以方便地在输入区中插入表情图片。

1. emoji.addActionListener(l-> {
2. picWindow.setVisible(true);//弹出表情框
3. });



1. public void insertSendPic(ImageIcon imgIc) {
2. content.insertIcon(imgIc); *// 插入图片*
3. }

##### 发送的处理

发送的处理较为麻烦。用户按下发送按键后先判空处理，然后开始组织json包。

先调用getFontAttrib函数获取输入区的内容和各个控件的选择返回一个FontAndText，内含发送文本，字体类型，字体大小，颜色等信息，将该FontAndText转成字符串加进json包的font字段

调用buildPicInfo获取表情信息。具体做法是通过输入区content的StyledDocument遍历每一位，判断是否为表情标签，如果是取出这张表情封装成PicInfo并加进myPicInfo数组中。且封装出表情信息串info返回，格式为“表情位置&表情序号+表情位置&表情序号+表情位置&表情序号……”如"8&3+12&11+"。将返回info装入json中的picInfo字段。

填写command和text内容后发送给服务器。

1. //返回文本样式信息
2. private FontAndText getFontAttrib() {
3. FontAndText att = new FontAndText();
4. att.setText(content.getText());*//文本信息*
5. att.setName((String) fontName.getSelectedItem());
6. att.setSize(Integer.parseInt((String) fontSize.getSelectedItem()));
7. String temp\_color = (String) fontColor.getSelectedItem();
8. if (temp\_color.equals("黑色")) {
9. att.setColor(new Color(0, 0, 0));
10. } else if (temp\_color.equals("红色")) {
11. att.setColor(new Color(255, 0, 0));
12. } else if (temp\_color.equals("蓝色")) {
13. att.setColor(new Color(0, 0, 255));
14. } else if (temp\_color.equals("黄色")) {
15. att.setColor(new Color(255, 255, 0));
16. } else if (temp\_color.equals("绿色")) {
17. att.setColor(new Color(0, 255, 0));
18. }
19. return att;
20. }
21. *//重组表情信息，重组后的信息串  格式为   位置&代号+位置&代号+……*
22. private String buildPicInfo() {
23. StringBuilder sb = new StringBuilder("");
24. *//遍历jtextpane找出所有的图片信息封装成指定格式*
25. for (int i = 0; i < this.content.getText().length(); i++) {
26. if (docContent.getCharacterElement(i).getName().equals("icon")) {
27. Icon icon = StyleConstants.getIcon(content.getStyledDocument().getCharacterElement(i).getAttributes());
28. ChatPic cupic = (ChatPic) icon;
29. PicInfo picInfo = new PicInfo(i, cupic.getIm() + "");
30. myPicInfo.add(picInfo);
31. sb.append(i).append("&").append(cupic.getIm()).append("+");
32. }
33. }
34. System.out.println(sb.toString());
35. return sb.toString();
36. }

#### 接收方接收信息时对表情的处理

接收线程接收到群发命令的json包，取出name,text,picInfo,font四个字段的数据，调用addRecMsg函数。

将name组合当前时间作为文本，加上font组成FontAndText类型的数据，调用insert函数插入聊天板中。（即每条信息前的用户 时间部分）

用getReceiveFont从font中拆解出信息，即将字符串font转换成FontAndText类型数据attr然后调用insert插入（即每条信息的正文部分）

如果检测到有表情信息，则调用receivedPicInfo将字符串picInfo拆解信息，将每个表情转换成PicInfo类型数据并存入receivePicInfo中。然后调用insertPics将receivePicInfo中的表情插入进文本中。注意receivePicInfo是缓存，每次用完要清空（即根据表情所在位置和序号在每条信息的正文部分中插入表情）

下面仅展示组织其他函数的addRecMsg函数：

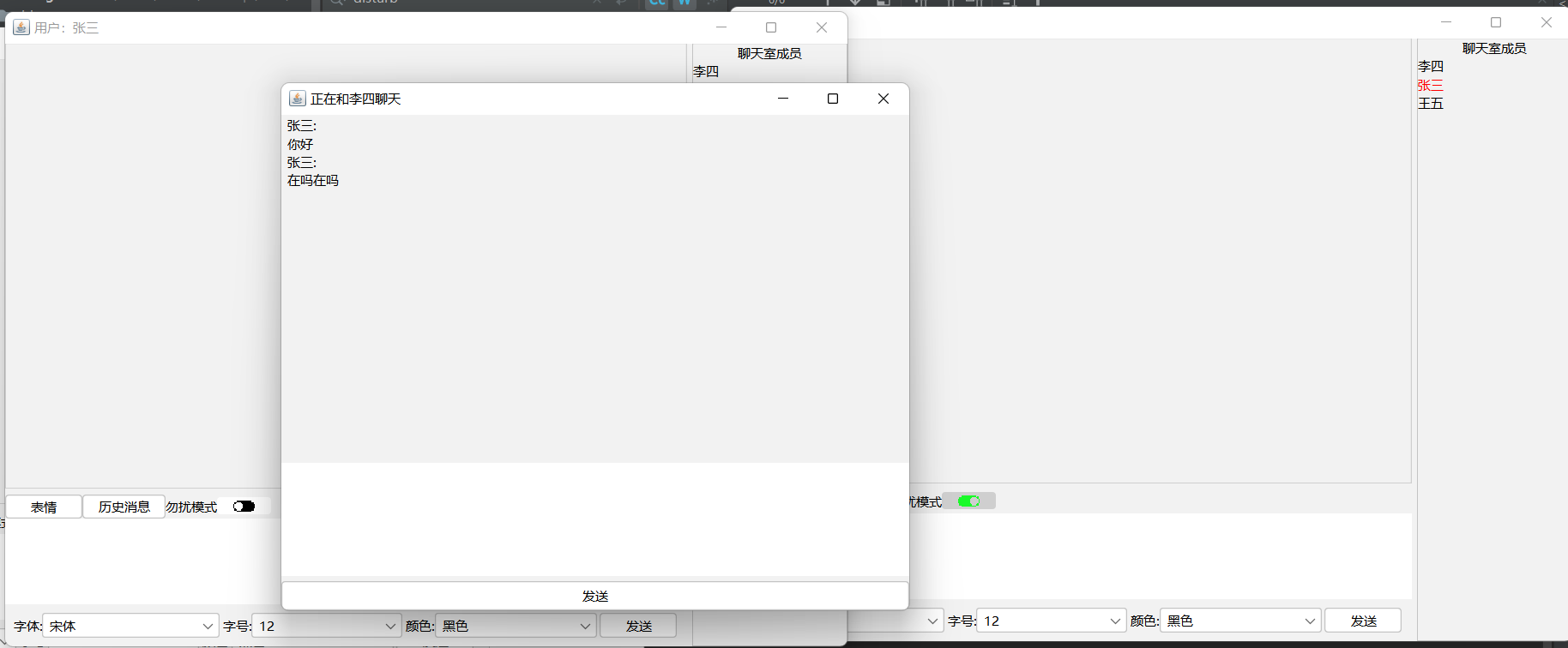
1. public void addRecMsg(String uname, String message,String picInfo,String font) {
2. String msg = uname + " " + sf.format(new Date());
3. dateFont.setText(msg);*//用户信息和时间*
4. insert(dateFont);
5. pos1 = chatBoard.getCaretPosition();*//记录插入的起始点*
6. if (!picInfo.equals("")) {*/\*存在表情信息\*/*
7. FontAndText attr = getReceiveFont(font);
8. insert(attr);
9. receivedPicInfo(picInfo);
10. insertPics();
11. } else {
12. FontAndText attr = getReceiveFont(font);
13. insert(attr);
14. }
15. }

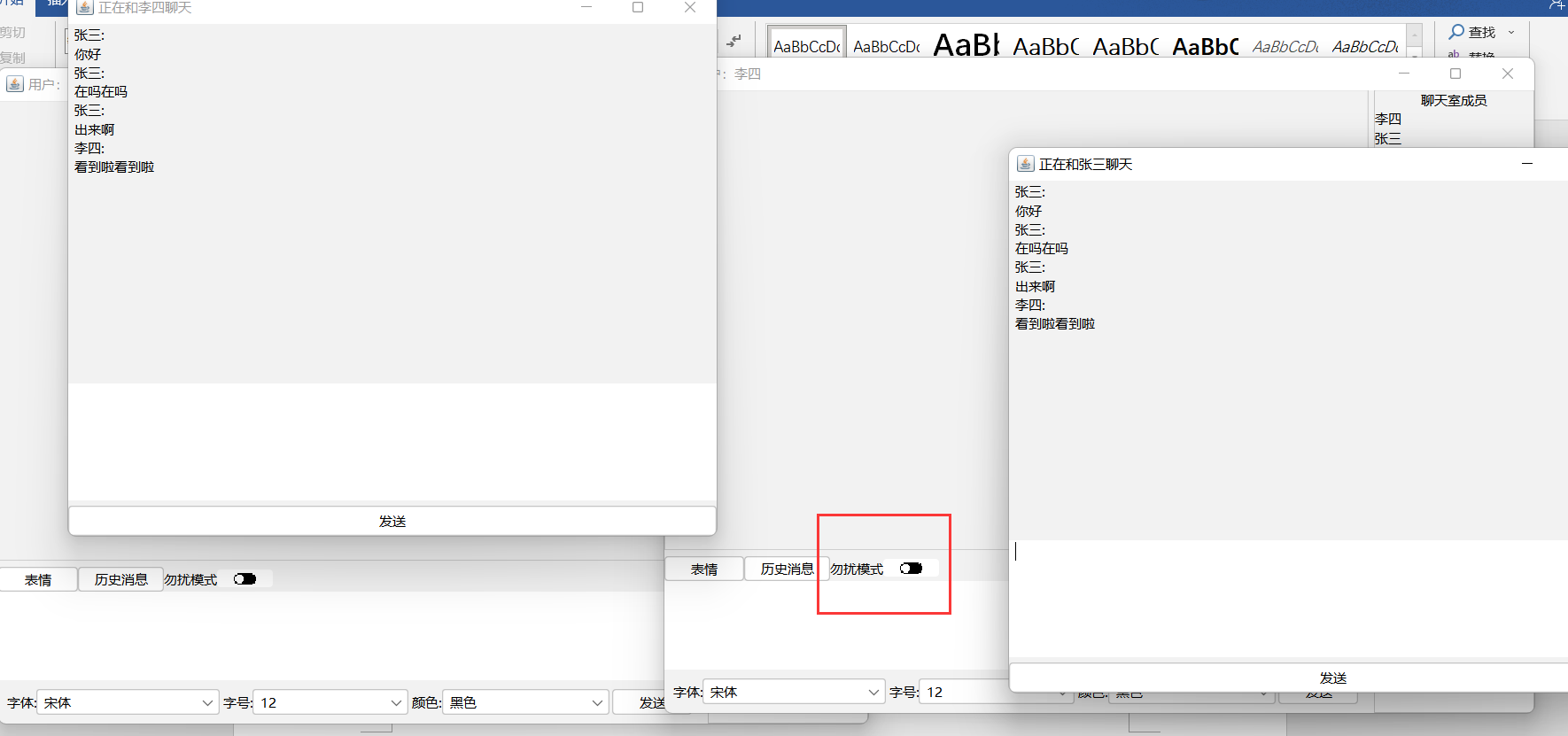
# 题目一的难点及解决

### 用户私聊信息的缓存（免打扰模式）

聊天室窗口右侧的在线成员列表中，双击标签即可弹出一个聊天对话框开始私人聊天，发送消息。考虑到如果每次有人发送私聊消息就弹出一个窗口，会对用户造成骚扰，于是增设了一个免打扰功能，开启免打扰功能后将发送方标签标红并缓存消息。这里使用Map<String,SingleChatFrame>singlewindow来管理该用户的所有私聊窗口，用Map<String,ArrayList<String>>messBuffer来管理所有用户发来的消息缓存。每当接收到私聊消息时取出发送方A，去singlewindow中找是否有A，如果有则直接在该聊天窗口添加内容。如果没有，在messBuffer的A上增加一条消息缓存，然后检查用户是否开了免打扰。如果开了免打扰，则将对方标红，结束。否则取出messBuffer中A的消息缓存并弹出和A的聊天窗口。

1. else if(mes==COMMAND\_SINGLE){
2. *//先检查singlewindow是否已经打开了窗口，如果打开了，控制该窗口添加消息*
3. String name=(String) receivedjsonObj.get("name");*//name是发送方的名字*
4. String text=(String) receivedjsonObj.get("text");*//text是对方发送的内容*
5. if(singlewindow.containsKey(name))
6. singlewindow.get(name).addMes(name,text);
7. *//如果还没打开窗口，先缓存，并将用户标红*
8. else{
9. if(!messBuffer.containsKey(name))
10. messBuffer.put(name,new ArrayList<>());
11. messBuffer.get(name).add(text);
12. for (JLabel ulbl : userLabelList)
13. if (name.equals(ulbl.getText()))
14. ulbl.setForeground(Color.red);
15. *//如果关闭勿扰模式则直接弹出弹窗*
16. if(!disturb.isSelected())
17. for(JLabel lbu:userLabelList)
18. if(lbu.getText().equals(name)){
19. singlewindow.put(name,new SingleChatFrame(name));
20. lbu.setForeground(Color.black);
21. }
22. }
23. }





### 登录/注册/注销时在线成员列表的动态刷新

用JPanel onlineUser存储在线成员列表，最开始简单地控制onlineUser的add和remove，发现更新没有效果，然后网上找了几个也会出现各种显示问题，比如直接白屏，最终找到了 panel.validate();panel.repaint();  panel.revalidate();试成功了，能够完成Jpanel的动态刷新。实现上登录/注册/注销都会提供要修改的用户名，在线用户名单存在Set<String> online中，直接对online进行修改，然后调用genMenberList函数，该函数每次都会遍历online重新绘制onlineUser。

void genMenberList(){

    menberListBox.removeAll();

    menberListBox.repaint();

    menberListBox.invalidate();

    userLabelList = new ArrayList<JLabel>();

    for (String us:online) {

        JLabel lbu = new JLabel(us);

        lbu.addMouseListener(new MouseAdapter() {

            @Override

            public void mouseClicked(MouseEvent e) {

*// 用户图标双击鼠标时显示对话框*

                if (e.getClickCount() == 2) {

                    singlewindow.put(us,new SingleChatFrame(us));

                    lbu.setForeground(Color.black);

                }

            }

        });

        userLabelList.add(lbu);

        menberListBox.add(lbu);

    }

    menberListBox.validate();

    menberListBox.updateUI();

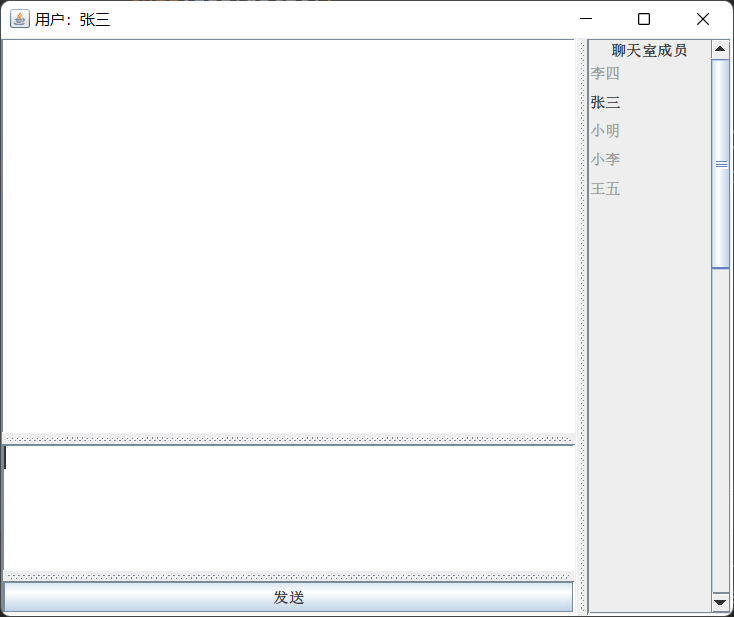
}

1. *//上下线刷新*
2. public void refreshFriendList(String userName, String flag) {
3. *// 更新聊天室成员列表*
4. if (flag.equals("1")) {
5. online.add(userName);
6. genMenberList();
7. }
8. else{
9. online.remove(userName);
10. genMenberList();
11. }
12. }

### 窗口放大缩小的动态变化

使用JSplitPane进行百分比分割时如果窗口放大缩小那么百分比分割的效果就会失效，经过搜索后添加监听事件监听窗口变化后部分解决。

1. this.addComponentListener(new ComponentAdapter(){
2. public void componentResized(ComponentEvent e) {
3. split.setDividerLocation(0.8);
4. left.setDividerLocation(0.7);
5. inputArea.setDividerLocation(0.8);
6. }
7. });

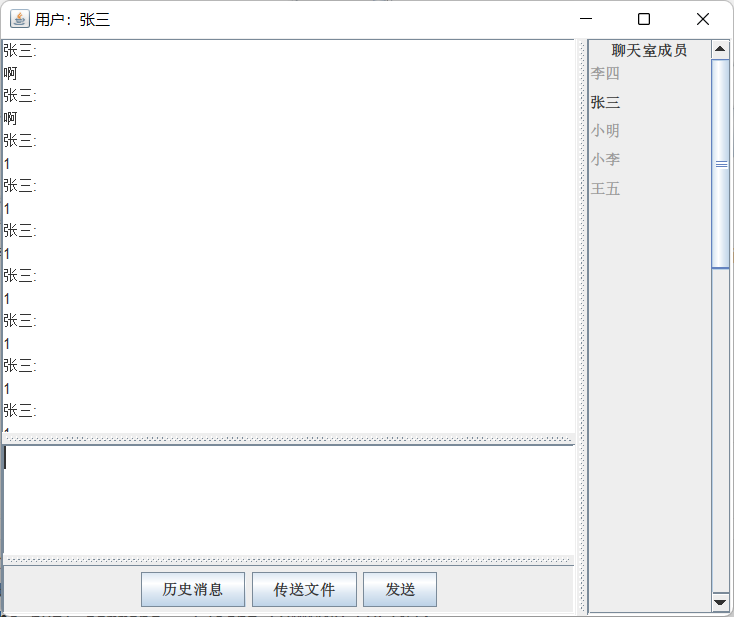
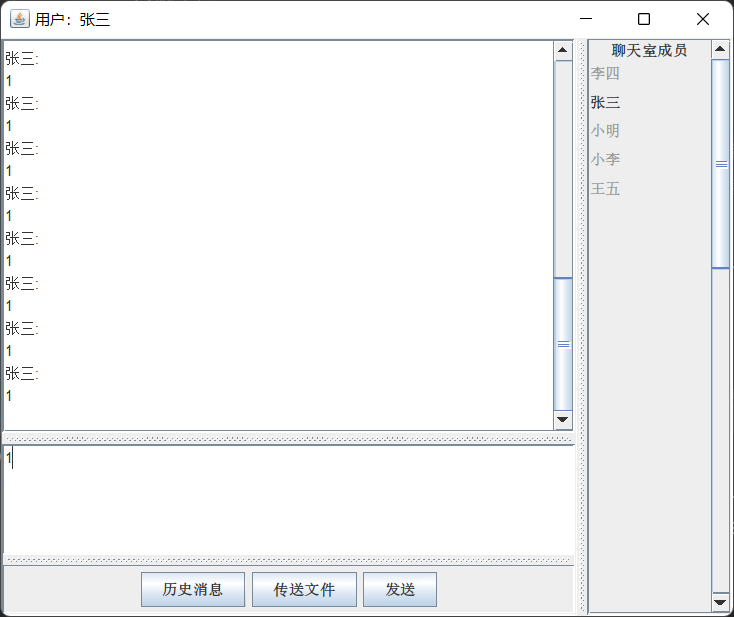


但放大后仍然有问题，后来经过尝试发现调换顺序后解决问题。

1. this.addComponentListener(new ComponentAdapter(){
2. public void componentResized(ComponentEvent e) {
3. inputArea.setDividerLocation(0.8);
4. split.setDividerLocation(0.8);
5. left.setDividerLocation(0.7);
6. }
7. });

### 聊天窗口的显示

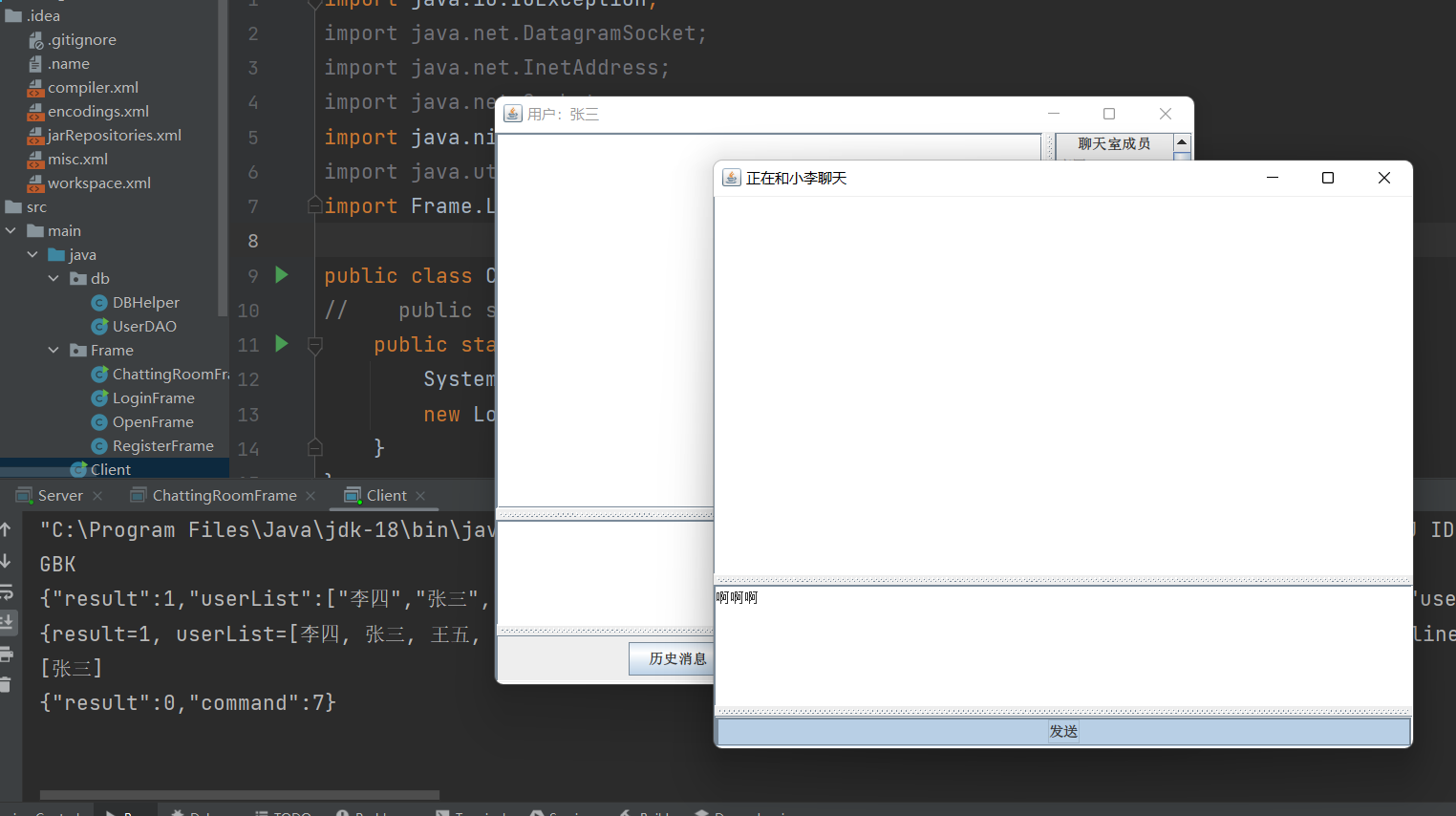
当输入多条信息时越界了聊天窗口中就看不到了

增设滚动条后得以解决。

### 私聊功能接收结果报文死机问题

为了共用一些变量我的私聊窗口和聊天区窗口放一个文件里了。私聊窗口中发送消息后需要接收结果报文，结果窗口死机了。

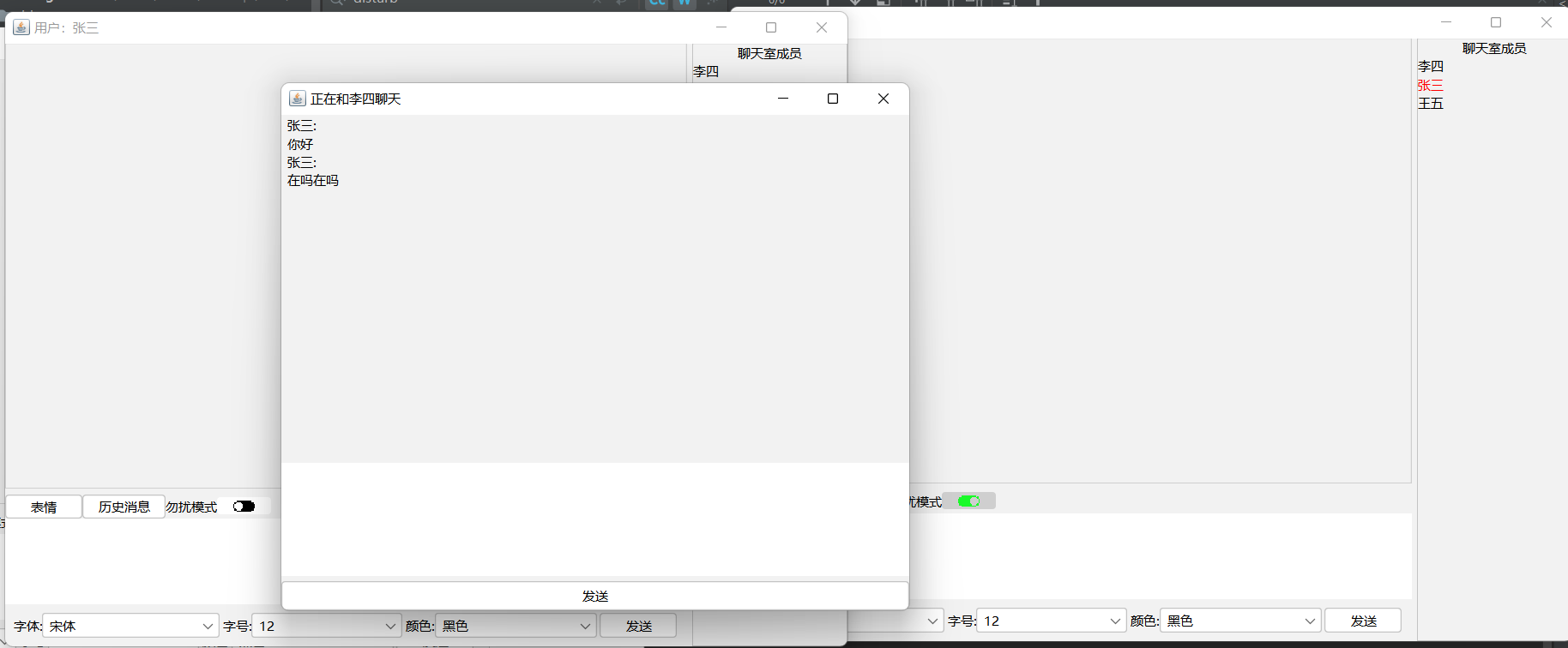


经排查发现由于私聊窗口和监听进程是并发的，所以私聊窗口等待的结果报文被聊天室窗口类中的独立的监听线程截获了，导致私聊窗口迟迟等不到回应报文死机了。这个想了挺久，最后是删去了私聊窗口等待结果报文的过程，结果报文由监听线程来接收，并调用私聊窗口的response接口来传递结果报文并更新私聊窗口。

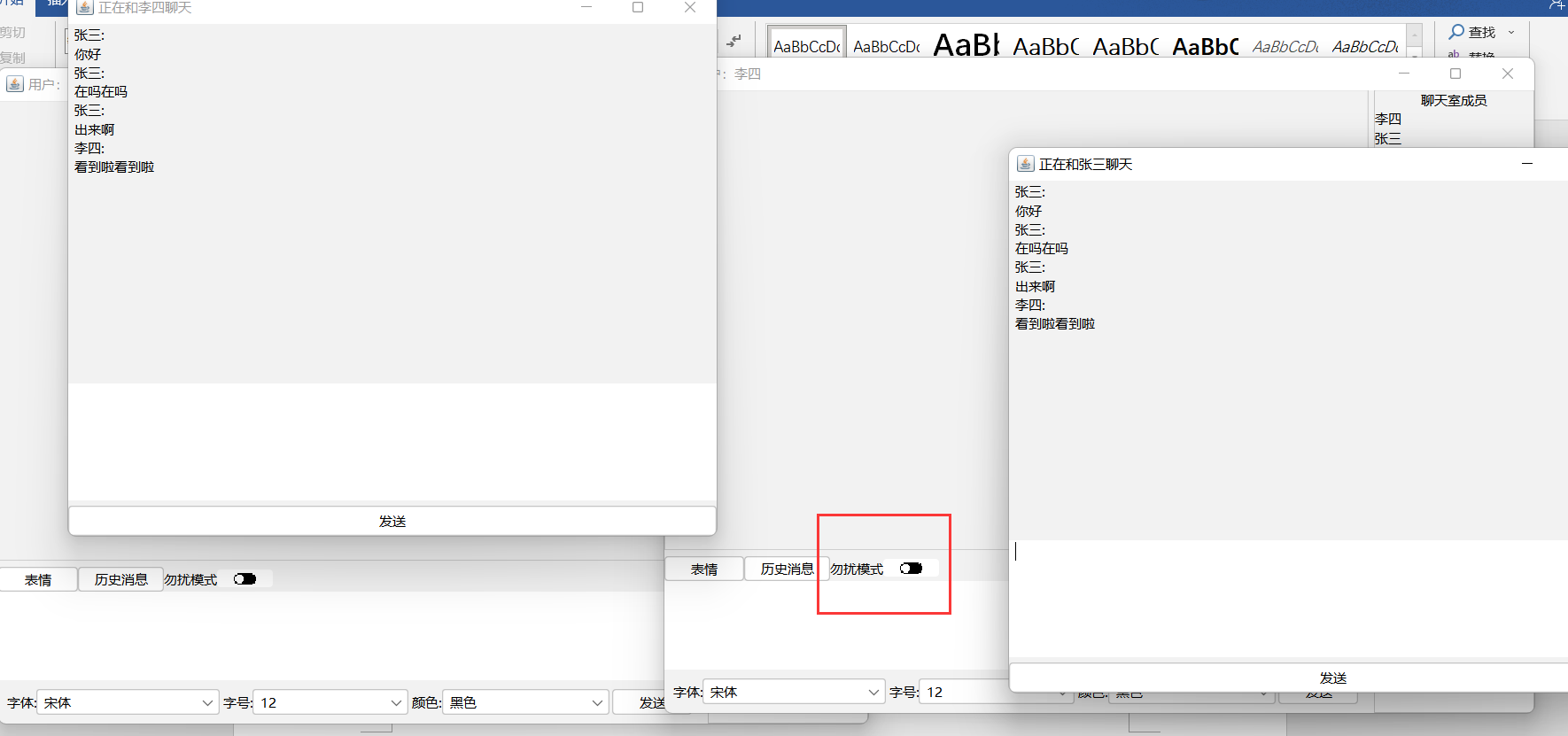
# 题目一的运行及测试结果

### 私聊功能

在免打扰模式下别人的私聊消息会进行缓存，并将发送方标红以提醒用户有新消息：

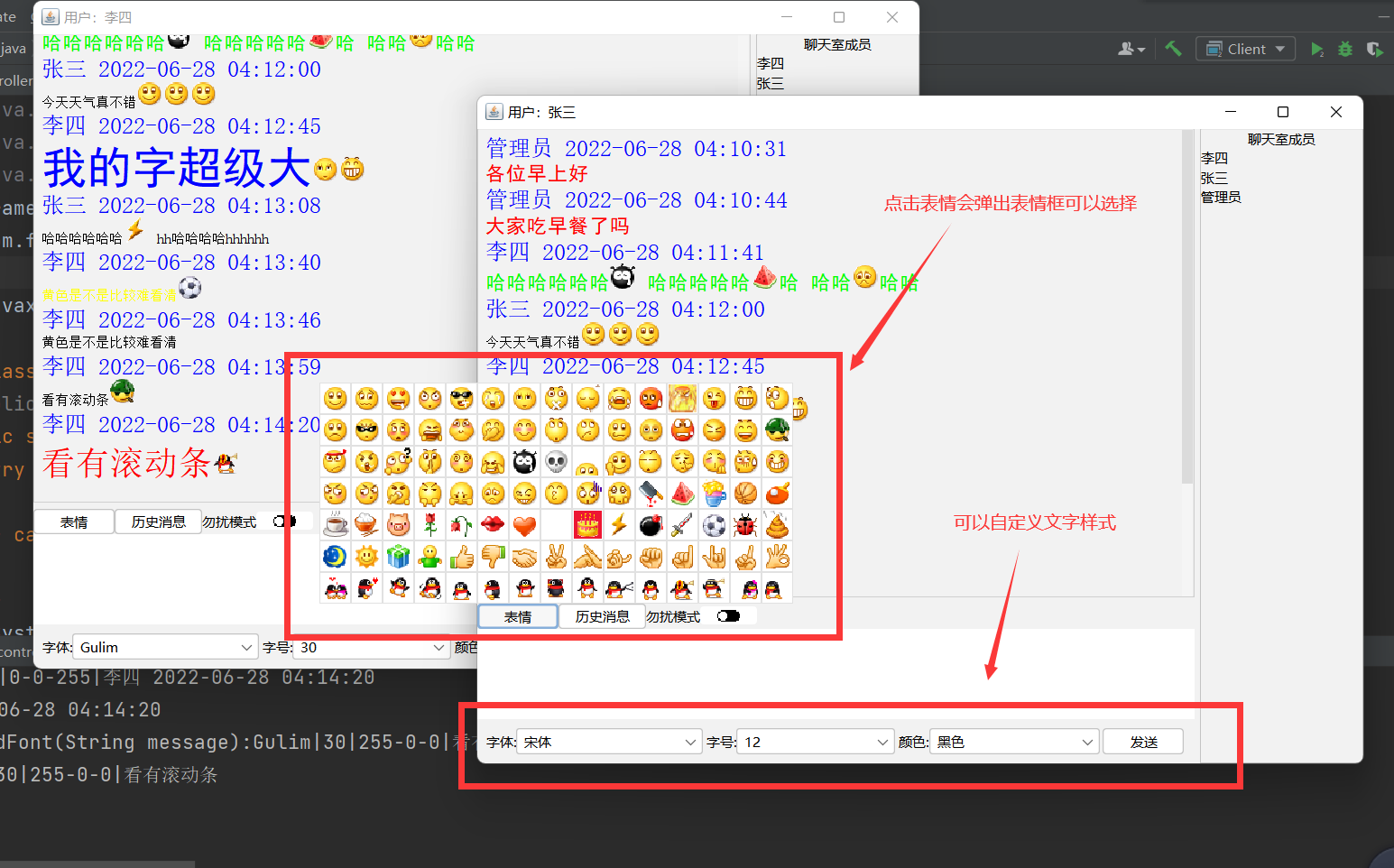


关闭免打扰后别人的私聊消息会直接弹出，如之前有消息会自动加载消息缓存：



### 聊天室群聊功能

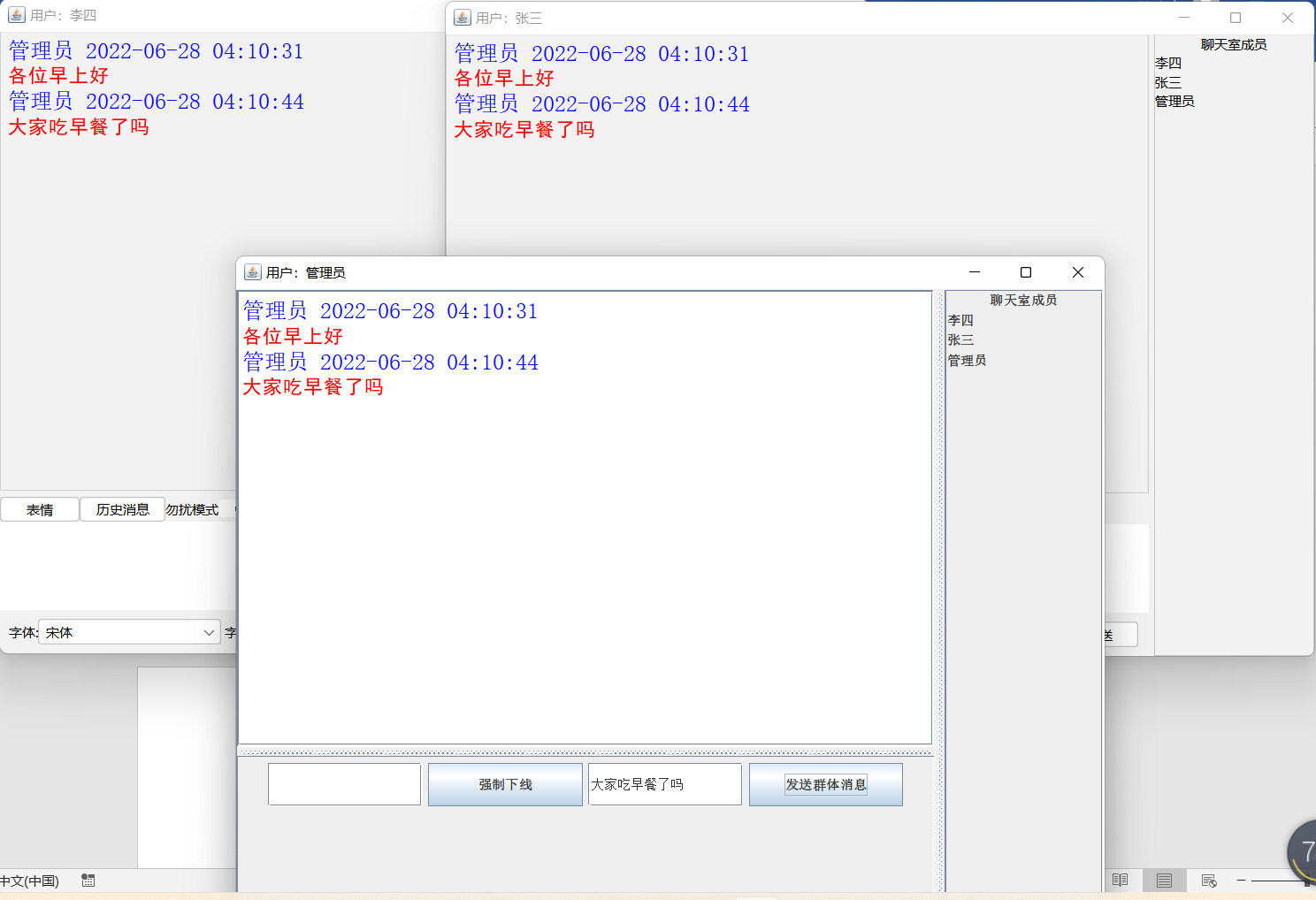
点击表情按键可以弹出表情框，发送区可自定义文字样式

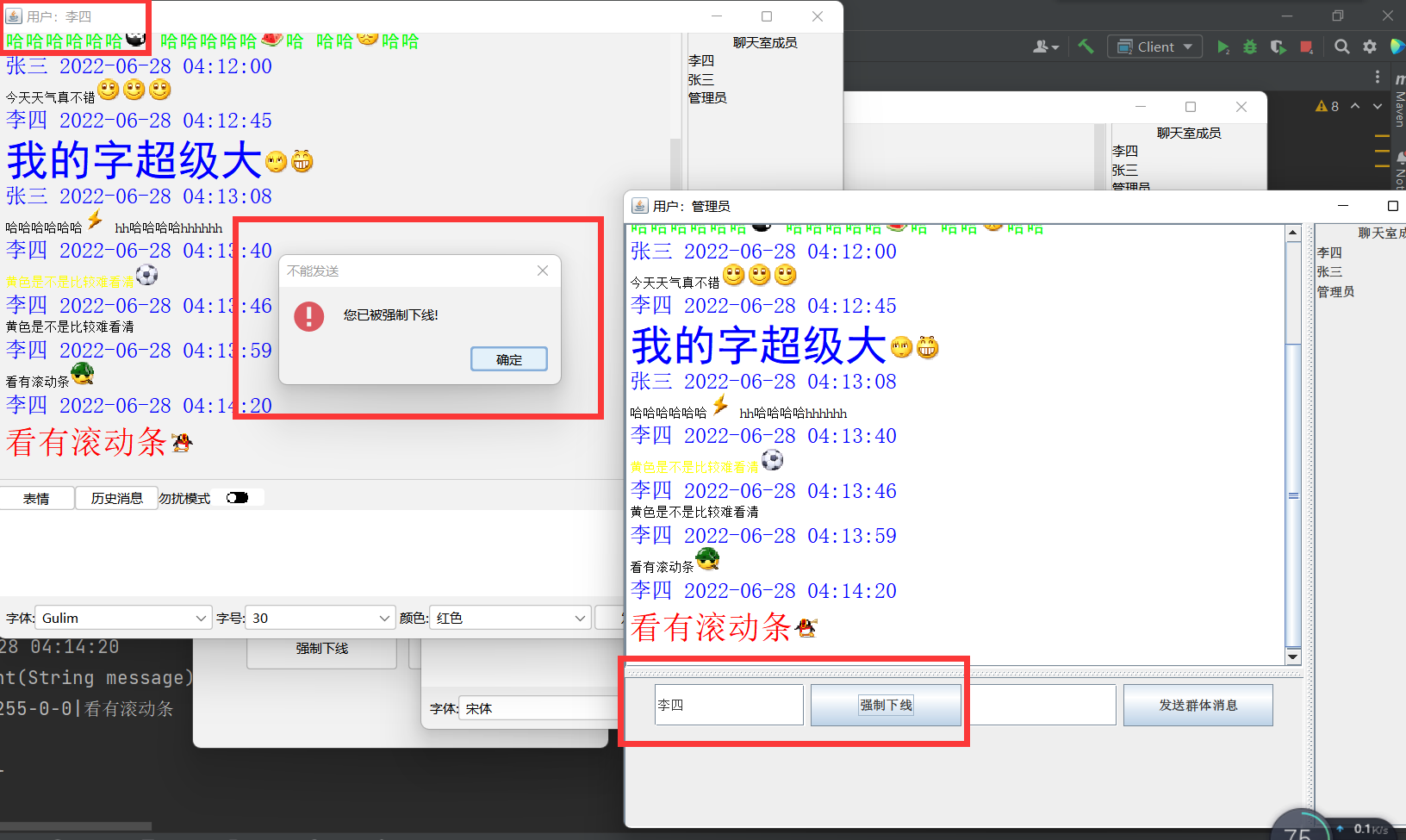


### 历史消息查询



### 服务器端





# 题目二要求

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。

# 题目二的设计思路

题目比较简单就只设置一个文件，需要实现的功能依次为：

1.爬取html源码，主要是URL类和URLConnection类的使用

2.从html源码中提取文本，这里采用正则表达式去除标签

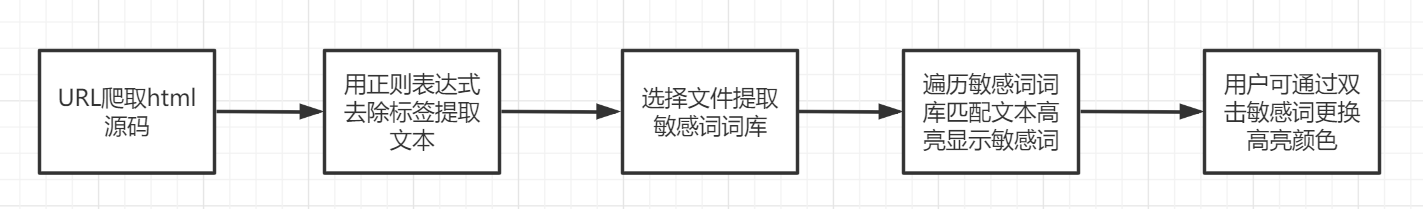
3.文本文件存储敏感词库，代码部分涉及文件读写，其中文件选择使用swing的JFileChooser。

4.提取文本敏感词并高亮显示，主要是涉及DefaultHighlighter的使用

5.文本文件存多个网址并爬取主要页数涉及文件读写和选择

6.这里考虑到所有敏感词都使用一个颜色看不出区别，增加一个双击敏感词给该敏感词更换颜色的功能。

# 题目二的实现过程



## 爬取html

这里主要是参考网上使用URL和URLConnection来爬取html文本的标准流程来写的，抽象类URLConnection是表示应用程序和URL之间的通信链接的所有类的超类。 该类的实例可以用于从URL引用的资源中读取和写入。获取输入流后读取每一行并添加进text中，最后返回html源代码text。

1. public String getHtml(String website) {
2. String str=null;
3. String text="";  *//保存网页的内容*
4. try {
5. URL url=new URL(website); *//建立对应的URL对象*
6. URLConnection urlConne=url.openConnection(); *//连接*
7. urlConne.connect();
8. *//获取输入流*
9. BufferedReader br=new BufferedReader(new InputStreamReader(urlConne.getInputStream(),"UTF-8"));
10. System.out.println("开始爬取");
11. while(true) { *//爬取到结束*
12. str=br.readLine();
13. if(str==null) break;
14. text+=(str+"\n");
15. }
16. br.close();  *//关闭输入流*
17. }catch (Exception e) {
18. *// TODO: handle exception*
19. JOptionPane.showMessageDialog(null, website+"爬取源代码失败");
20. }
21. System.out.println("爬取结束");
22. return text; *//返回html代码文本*
23. }

## 提取文本

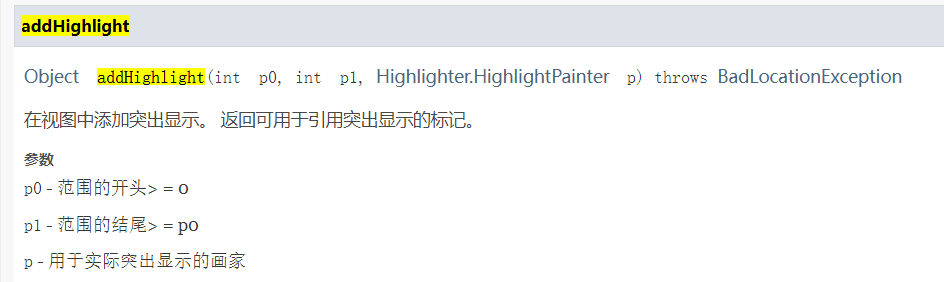
提取文本主要是使用正则表达式匹配，注意到script是代码标签，style是样式标签，这两种标签中间的内容是不需要的，所以先去除这两种标签以及它们中间的内容，然后去除剩下的标签（其他标签中间的内容要保留，这些就是需要提取的文本）。要注意正则表达式匹配的顺序不可以调换，如果提前去除标签就找不到style和script标签了，代码和样式也会保留，而这些是不需要的。

1. *//设置正则表达式的匹配符*
2. private String regExHtml="<[^>]+>";  *//匹配标签*
3. private String regExScript = "<script[^>]\*?>[\\s\\S]\*?<\\/script>";  *//匹配script标签*
4. private String regExStyle = "<style[^>]\*?>[\\s\\S]\*?<\\/style>";  *//匹配style标签*
5. private String regExSpace="[\\s]{2,}"; *//匹配连续空格或回车等*
6. private String regExImg="&[\\S]\*?;+"; *//匹配网页上图案的乱码*
7. *//定义正则表达式*
8. private Pattern pattern3=Pattern.compile(regExHtml, Pattern.CASE\_INSENSITIVE);
9. private Pattern pattern1=Pattern.compile(regExScript,Pattern.CASE\_INSENSITIVE);
10. private Pattern pattern2=Pattern.compile(regExStyle,Pattern.CASE\_INSENSITIVE);
11. private Pattern pattern4=Pattern.compile(regExSpace, Pattern.CASE\_INSENSITIVE);
12. private Pattern pattern5=Pattern.compile(regExImg,Pattern.CASE\_INSENSITIVE);
13. *//对html进行正则匹配,提取出其中的文本*
14. public String getText(String str) {
15. Matcher matcher=pattern1.matcher(str);
16. str=matcher.replaceAll("");  *//匹配普通标签*
17. matcher=pattern2.matcher(str);
18. str=matcher.replaceAll("");  *//匹配script标签*
19. matcher=pattern3.matcher(str);
20. str=matcher.replaceAll("");  *//匹配style标签*
21. matcher=pattern4.matcher(str);
22. str=matcher.replaceAll("\n"); *//匹配连续回车或空格*
23. matcher=pattern5.matcher(str);
24. str=matcher.replaceAll("");  *//匹配网页图案出现的乱码*
25. return str;  *//返回文本*
26. }

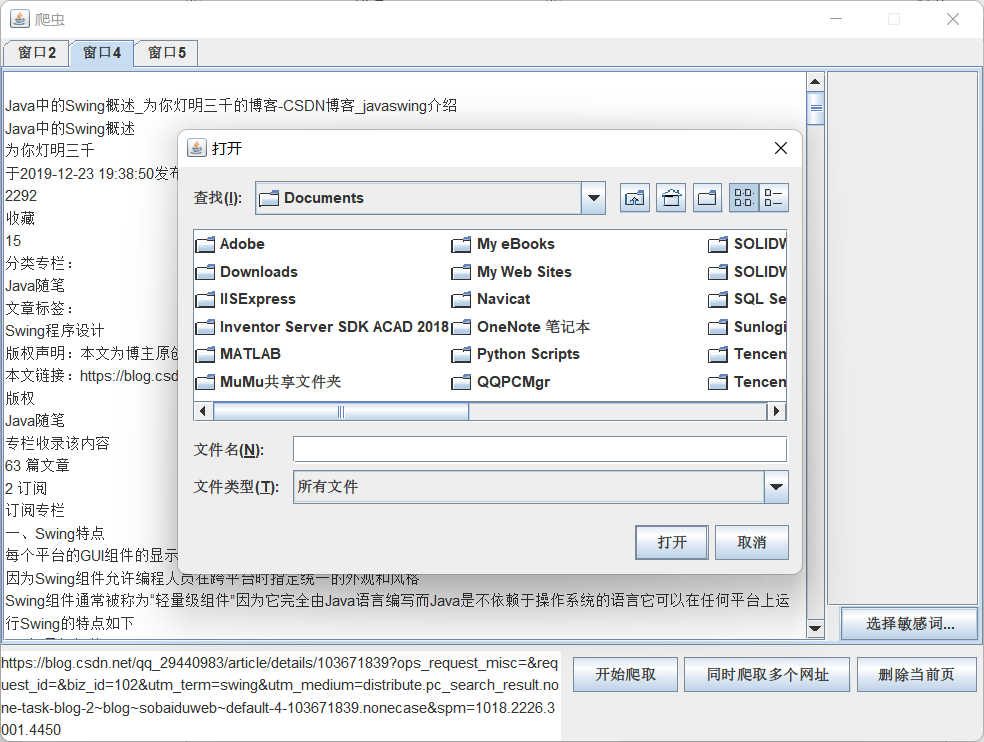
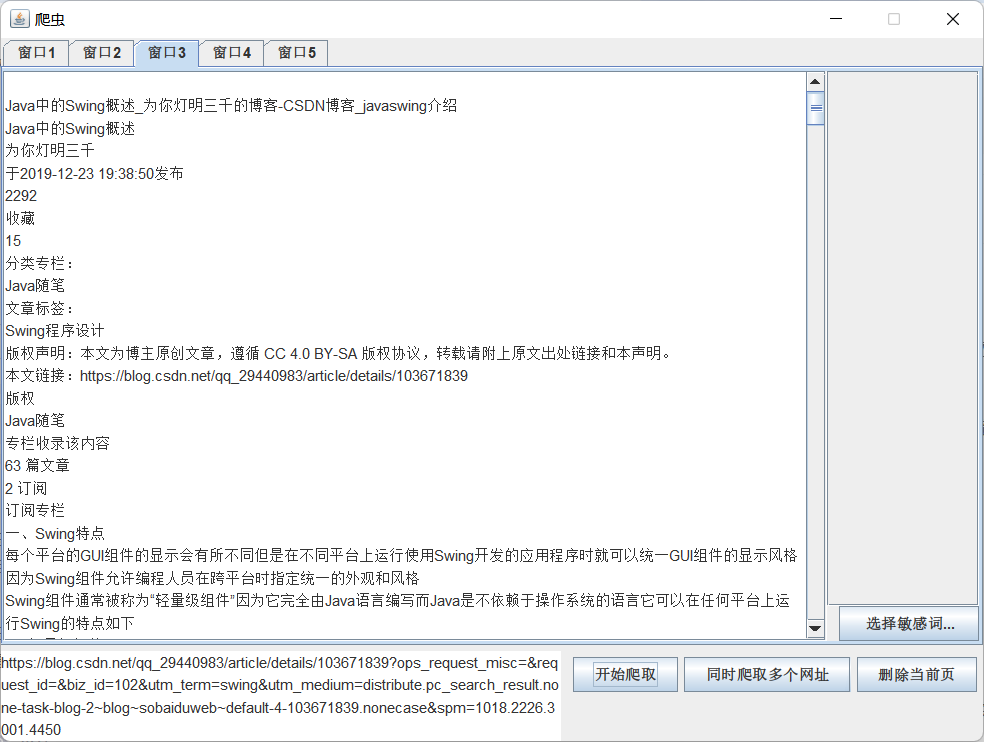
## 界面设计和高亮显示

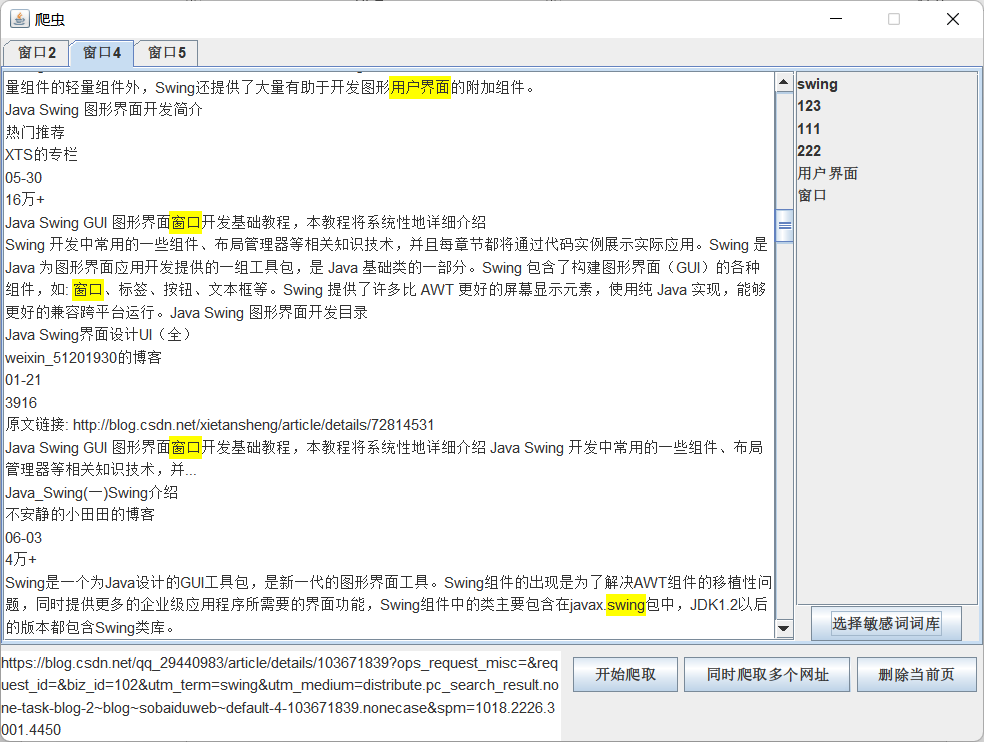
界面上一个输入区，包含一个输入框和一些按钮（开始爬取，爬取多个网址，删除一个结果页），一个显示区显示爬取的文本内容，一个侧栏显示敏感词。展示结果时多窗口会混乱，但爬过的结果由需要保留，为了方便起见，展示结果用JTabbedPane选项卡组件，将一次结果封装成一个page类，继承自JPanel，每产生一次结果就建一个page并添加进选项卡中，设置删除按钮点击时获取当前页面并删除，即通过tab.remove(tab.getSelectedIndex())删除。

Page页接收爬取的文本并展示在JTextArea上，每个结果都可以选择自己的敏感词词库，所以每个page页都要有一个选择敏感词库的按键，通过JFileChooser选择文件读取敏感词存入wordList中。高亮显示时遍历wordList取出每一个敏感词，用indexOf在文本串中不断匹配该敏感词，用hg.addHighlight(index, index+str.length(), painter)添加高亮。其中addHighlight的用法如下



1. public void showSensword() {
2. Highlighter hg=txt.getHighlighter(); *//设置文本框的高亮显示*
3. hg.removeAllHighlights(); *//清除之前的高亮显示记录*
4. String text=txt.getText(); *//得到文本框的文本*
5. DefaultHighlighter.DefaultHighlightPainter painter=new DefaultHighlighter.DefaultHighlightPainter(Color.YELLOW); *//设置高亮显示颜色为黄色*
6. *//对于每一个敏感词，用while循环遍历整个字符串找到匹配到的每一个位置，在每一个位置上增加一个高亮*
7. for(String str:wordList) { *//匹配其中的每一个敏感词*
8. ArrayList<Object>tmp=new ArrayList<>();
9. int index=0,cnt=0;
10. while((index=text.indexOf(str,index))>=0) {
11. try {
12. tmp.add(hg.addHighlight(index, index+str.length(), painter));*//高亮显示匹配到的词语*
13. index+=str.length(); *//更新匹配条件继续匹配*
14. cnt++;
15. } catch (BadLocationException e) {
16. *// TODO Auto-generated catch block*
17. e.printStackTrace();
18. }
19. }
20. hightLight.put(str,tmp);
21. ColorState.put(str,0);*//初始都为黄色*
22. System.out.println(cnt);
23. count.put(str,(Integer) cnt);
24. }
25. }





## 爬取多个网站

新建SpiderAll继承自Thread类，以便爬取文件中的多个网站的同时用户还能查询其他网站。调用前先用两个JFileChooser选择敏感词词库文件和存URL的文件，读取敏感词词库进入全局变量wordList中，然后和爬取一个网站的流程相同，调用getHtml获取html源代码，调用getText从源码中提取文本，遍历wordList取出每一个敏感词和文本进行匹配，每匹配到一个wordNum中该敏感词对应位+1。最终整合数据并写入文件即可。

1. public void run() {
2. try {
3. *//读取网址库中的网址*
4. BufferedReader brr=new BufferedReader(new FileReader(file));
5. *//将匹配数据写入文本中*
6. PrintStream ps=new PrintStream(new File("data.txt"));
7. ps.println("敏感词记录如下:");
8. int size=wordList.size();
9. while(true) {
10. String website=brr.readLine();
11. if(website==null) break;
12. ps.println(website+"数据如下: ");
13. String html=getHtml(website); *//获取html代码*
14. String text=getText(html);  *//匹配网页文本*
15. for(int i=0;i<size;i++) {  *//在网页文本中进行匹配*
16. String word=wordList.get(i);
17. int index=0,account=0,len=word.length();
18. while((index=text.indexOf(word,index))>=0) {
19. account++;
20. int temp=wordNum.get(i); *//更新数据*
21. wordNum.set(i,++temp);
22. index+=len;  *//更新匹配条件*
23. }
24. ps.println(word+"  出现  "+account+"次"); *//写入当前数据*
25. }
26. ps.println();
27. }
28. brr.close(); *//关闭文件流*
29. System.out.println("爬取完毕");
30. ps.println("总数据如下:     ");  *//写入总数据*
31. for(int i=0;i<size;i++) {
32. ps.println(wordList.get(i)+"  出现    "+wordNum.get(i)+"次");
33. }
34. ps.close();  *//关闭文件流*
35. JOptionPane.showMessageDialog(null, "爬取完毕！请打开文件查看!");
36. }catch (Exception e) {
37. *// TODO: handle exception*
38. JOptionPane.showMessageDialog(null, "爬取失败");
39. }
40. }

# 题目二的难点及解决

## 给敏感词的高亮换色

考虑到所有敏感词都使用一个颜色看不出区别，为侧栏的每个敏感词标签绑定一个双击事件，每次双击时就给该敏感词对应的高亮换一种颜色。

每个page页面中用HashMap<String,ArrayList<Object>>hightLight存储每个敏感词的高亮对象数组。高亮标出敏感词时用的是tmp.add(hg.addHighlight(index, index+str.length(), painter)); 然后hightLight.put(str,tmp);这样就把所有敏感词对应的高亮对象存储进hightLight中了。HashMap<String,Integer>ColorState记录每个敏感词对应的颜色，用Int来代表颜色，总共4种，即每次换颜色就是+1模4再去找对应的颜色。绑定双击事件如下：

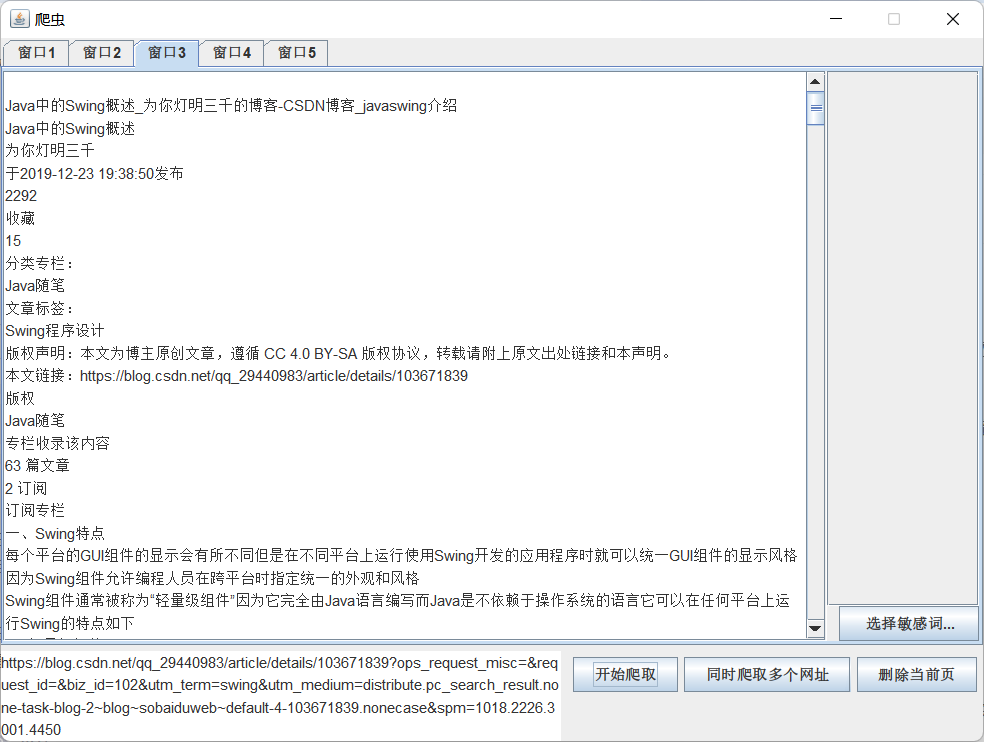
1. jbl.addMouseListener(new MouseAdapter() {
2. @Override
3. public void mouseClicked(MouseEvent e) {
4. *// 双击敏感词换色。根据点击标签的文字找到该关键字对应的那几个高亮对象并一一删除*
5. *// 在文本中重新对该关键字上色，用不同颜色。*
6. if (e.getClickCount() == 2) {
7. String s=jbl.getText();
8. System.out.println(s);
9. Highlighter hg=txt.getHighlighter();*//获取文本总的高亮对象*
10. ArrayList<Object>arr=hightLight.get(s);
11. for(Object obj:arr){
12. hg.removeHighlight(obj);
13. }
14. *//获得下一个颜色重新绘制*
15. ColorState.put(s,(ColorState.get(s)+1)%ColorTrans.size());
16. DefaultHighlighter.DefaultHighlightPainter painter=new
17. DefaultHighlighter.DefaultHighlightPainter(ColorTrans.get(ColorState.get(s))); *//设置高亮显示颜色为黄色*
18. *//重新绘制高亮*
19. ArrayList<Object>tmp=new ArrayList<>();
20. int index=0;
21. String text=txt.getText(); *//得到文本框的文本*
22. while((index=text.indexOf(str,index))>=0) {
23. try {
24. tmp.add(hg.addHighlight(index, index+str.length(), painter));*//高亮显示匹配到的词语*
25. index+=str.length(); *//更新匹配条件继续匹配*
26. } catch (BadLocationException i) {
27. *// TODO Auto-generated catch block*
28. i.printStackTrace();
29. }
30. }
31. hightLight.put(str,tmp);
32. }
33. }
34. });



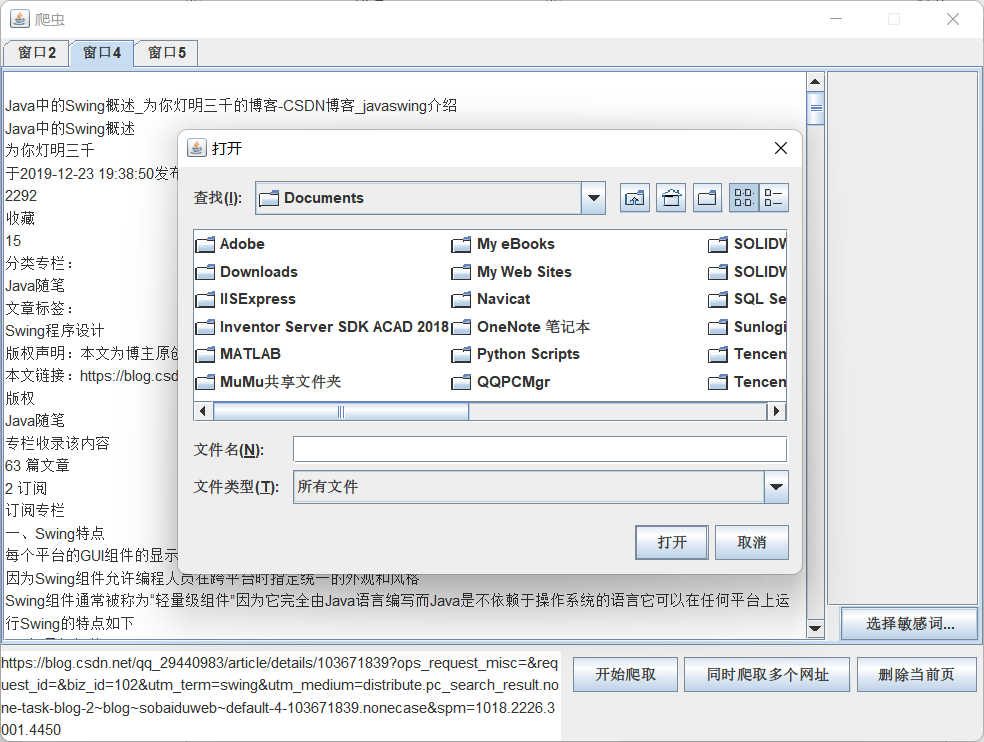
7双击换色效果

# 题目二的运行及测试结果

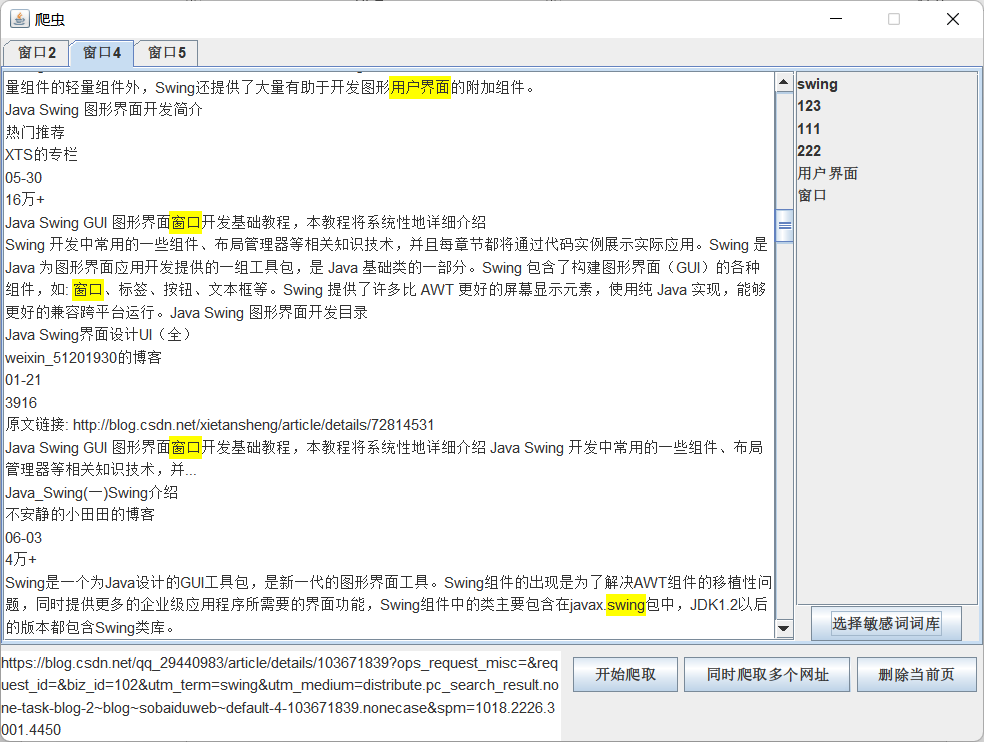
## 单个网页爬取效果图







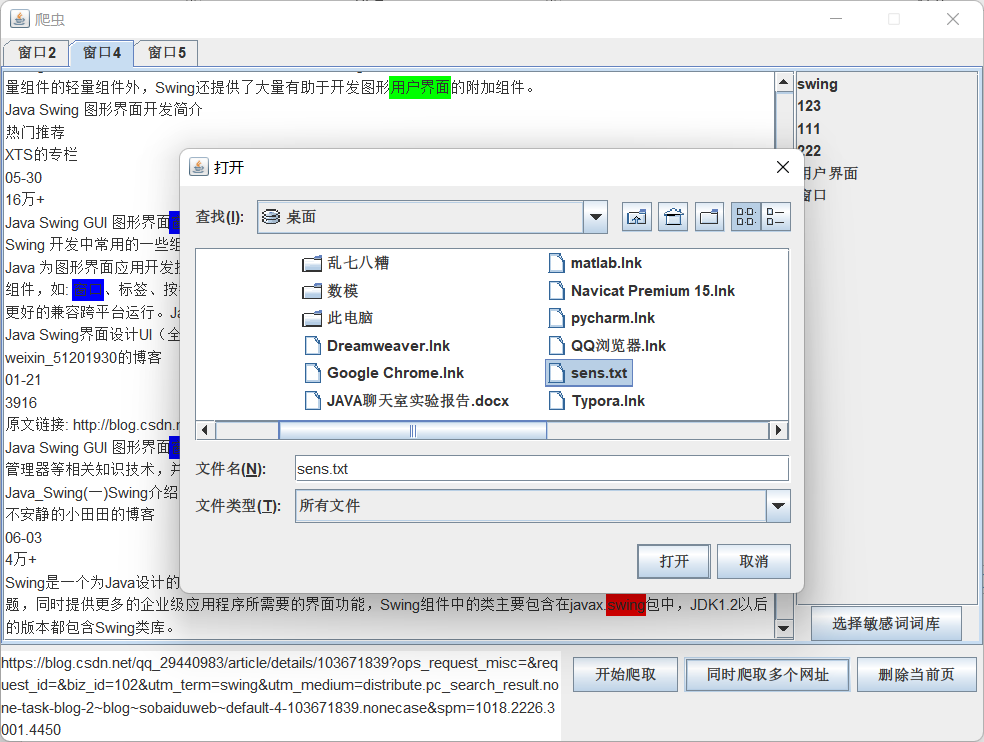
8选择敏感词



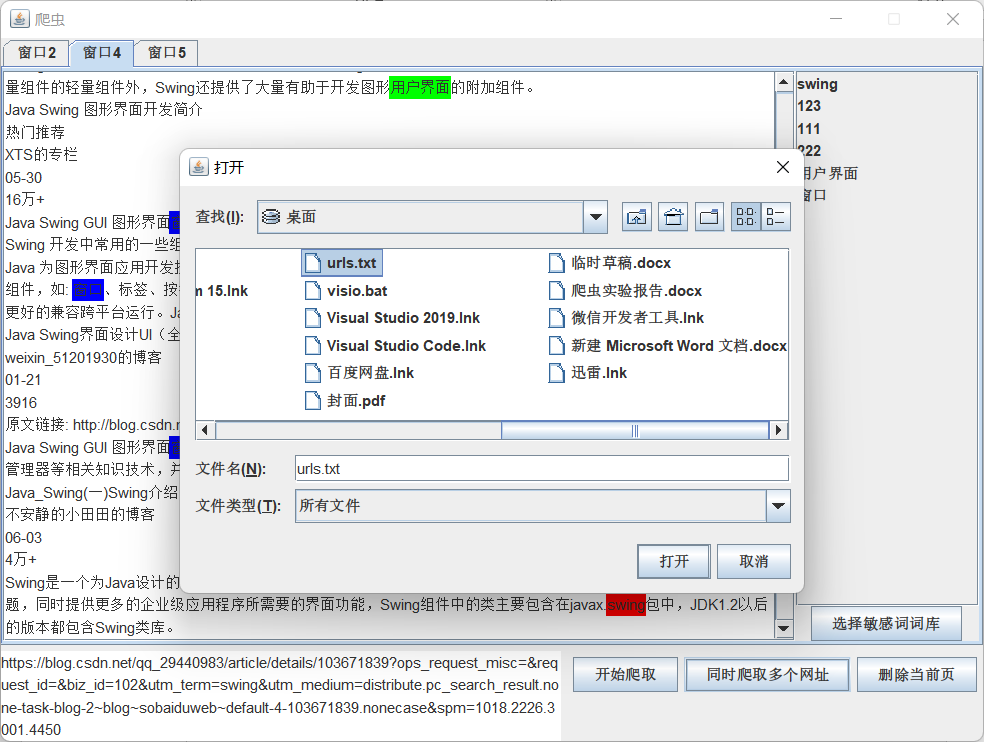


9双击更换颜色

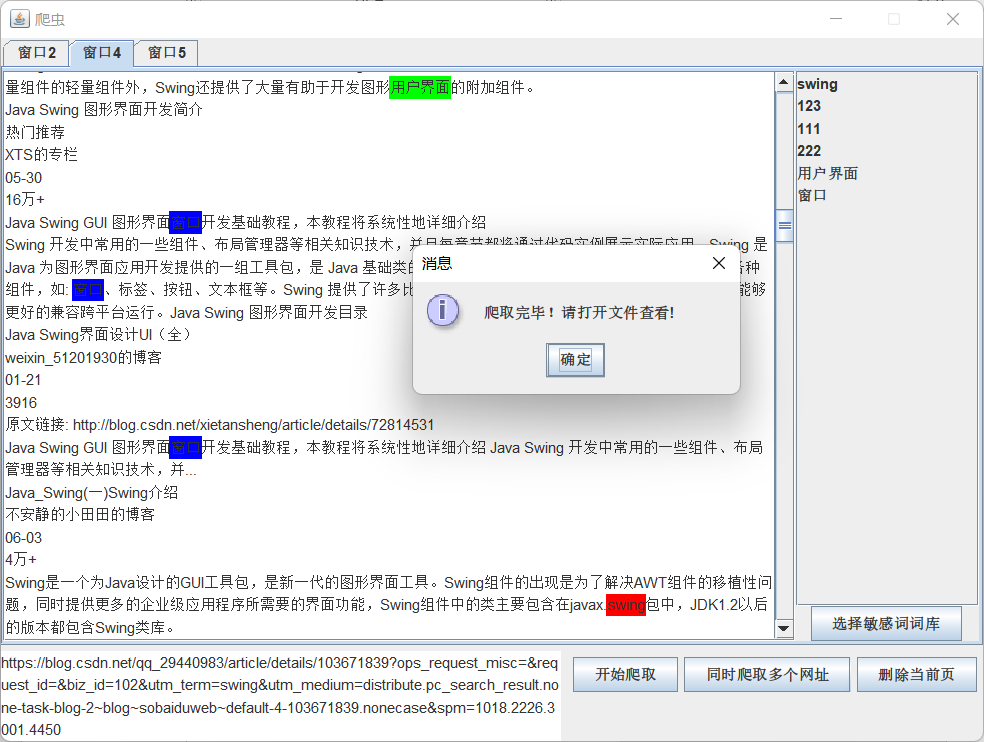
## 爬取多个网站

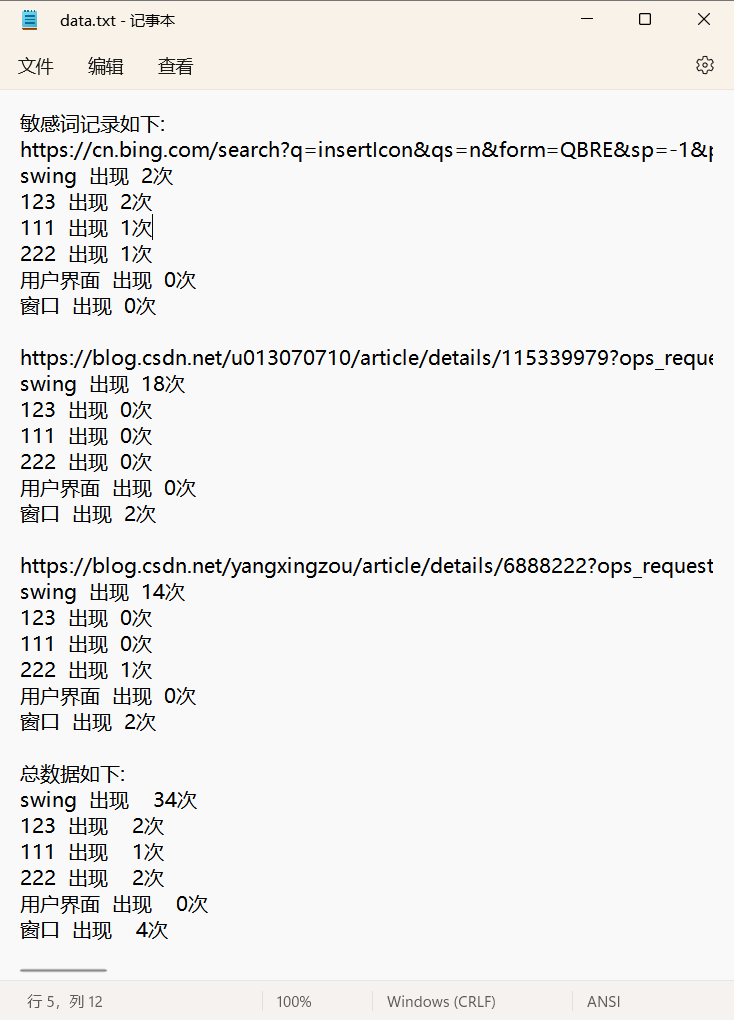


10选择敏感词库



11选择URL文件





12爬取结果写入文本文件

# 总体心得体会

由于上学期没有学JAVA，这次是边做边学的，这算是我写的第一个java程序了。感觉是从零开始学一路跌跌撞撞做了很久，很多时候辛辛苦苦写了一大片然后发现有很多更好的做法然后又改了一大片。总得来说收获非常大，从JAVA的基本语法和基本容器的使用到用JAVA连接数据库，JSON传输，可视化编程，多线程编程，时间日期操作，网络通信SOCKET编程等很多方面的知识都有涉及到，学到了非常多。爬虫项目也是熟悉了URL类和URLconnections类和正则表达式的使用，以及文件流读取，字符串匹配等相关知识。同时也加强了图形化界面编程的训练，通过继承自定义控件，选项卡组件的使用等也更熟悉了。这是我这学期付出最多时间和精力的的一个项目了，看着程序从功能简陋界面粗糙结构混杂一点点变得功能齐全界面看得过眼程序健壮性也还可以的的程度，还是充满成就感的。