****

**应用基础实践（一）网络与JAVA**

**课程设计实验报告**

|  |  |
| --- | --- |
| 学生姓名 | 张子洋 |
| 学 号 | 808221223 |
| 专业班级 | 计科2203班 |
| 指导教师 | 刘嫔 |
| 学 院 | 计算机学院 |
| 完成时间 | 2024.7.5 |

计算机学院

2024年7月

目录

[一、 系统描述 3](#_Toc25192)

[（一）用JavaSocket编程开发聊天室 3](#_Toc16504)

[（二）用JavaURL编程爬取并分析网页敏感词 3](#_Toc11467)

[二、 功能模块结构 4](#_Toc9594)

[实验一 4](#_Toc20197)

[实验二 5](#_Toc29424)

[三、 主要模块的算法说明 9](#_Toc21981)

[实验一 9](#_Toc28698)

[服务器端 (Server) 9](#_Toc1988)

[客户端 (Client) 10](#_Toc24455)

[实验二 11](#_Toc32308)

[1. 界面模块 11](#_Toc26005)

[2. 网页爬取模块 11](#_Toc18149)

[3. 敏感词处理模块 11](#_Toc8716)

[四、 运行结果 12](#_Toc769)

[实验一 12](#_Toc31195)

[实验二 14](#_Toc13648)

[五、 课程设计总结 16](#_Toc30401)

[课程设计中的问题、解决过程及体会 16](#_Toc30710)

[课程设计的收获 17](#_Toc17960)

[对程序开发的认识与思考 17](#_Toc401)

[六、 参考文献 18](#_Toc19207)

[七、 附录 19](#_Toc19372)

[实验一 19](#_Toc32721)

[实验二 34](#_Toc22819)

# 系统描述

（一）用JavaSocket编程开发聊天室

实践基本要求：

1.了解实验目的及实验原理；

2.编写程序，并附上程序代码和结果图；

3.总结在编程过程中遇到的问题、解决办法和收获。

实践的内容或要求：

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。

7. 用户能够自己建立小群聊天（选做）并解散小群。

（二）用JavaURL编程爬取并分析网页敏感词

实践基本要求：

1.了解实验目的及实验原理；

2.编写程序，并附上程序代码和结果图；

3.总结在编程过程中遇到的问题、解决办法和收获。

实践的内容或要求：

1.编写设计界面，输入一个网址，能够持续爬取从该地址开始链接的所有相关网页，并能控制爬取工作的开始和停止。

2.对爬取网页中的文本进行提取。

3.建立敏感词库，用文本文件保存，能够修改更新该敏感词库。

4.将爬取网页的文本中的敏感词提取出来并高亮显示。

5．为所有爬取的网页建立可视化有向图网（选做）。

6.编写一个主界面,整合上述功能。

# 功能模块结构

## 实验一

### ****系统结构设计：****

**客户端：**

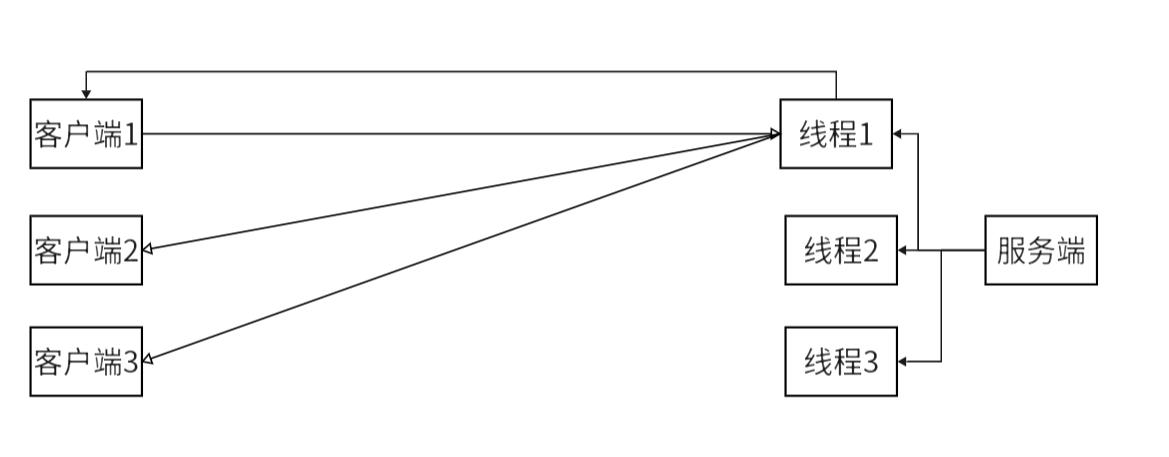
* 客户端提供一个图形用户界面（GUI），供用户输入他们想要访问的网址。用户界面通过Java Swing实现，包括一个文本框用于输入网址和一个提交按钮用于发送请求。
* 客户端通过Socket编程与服务器进行通信。客户端将用户输入的网址发送到服务器以进行验证，并等待服务器返回结果。

**服务器端：**

* 服务器端维护一个黑名单，包含禁止访问的IP地址。这些IP地址可以存储在文件或数据库中，以便动态加载和更新。
* 服务器端使用Socket编程接收客户端的请求，获取请求的IP地址，并检查该IP地址是否在黑名单中。检查结果决定是否允许客户端访问请求的网址。
* 服务器端将检查结果发送回客户端。

### ****工作流程：****

* 用户在客户端输入网址并提交请求。客户端将请求发送到服务器。
* 服务器接收到客户端请求后，提取出请求的网址并获取对应的IP地址。
* 服务器检查该IP地址是否在黑名单中：
* 如果IP地址在黑名单中，服务器拒绝请求并返回拒绝访问的消息给客户端。
* 如果IP地址不在黑名单中，服务器允许请求并返回允许访问的消息给客户端。
* 客户端接收服务器的返回消息，并在用户界面上显示结果。



### ****技术实现：****

**客户端：**

* 使用Java Swing构建用户界面，提供输入网址的文本框和提交按钮。用户界面包括基本的布局管理和事件处理。
* 使用Socket编程实现客户端与服务器的通信。客户端通过Socket连接到服务器，并发送用户输入的网址。

**服务器端：**

* 使用Socket编程实现服务器端接收客户端请求。服务器创建一个ServerSocket，监听客户端的连接请求。
* 服务器维护一个黑名单，可以使用文件或数据库存储黑名单IP地址。服务器启动时加载黑名单，并定期检查和更新。
* 实现IP地址检查逻辑。服务器接收到客户端请求后，提取出IP地址并与黑名单进行匹配，决定是否允许访问。

## 实验二

### 系统结构设计

**主窗口**

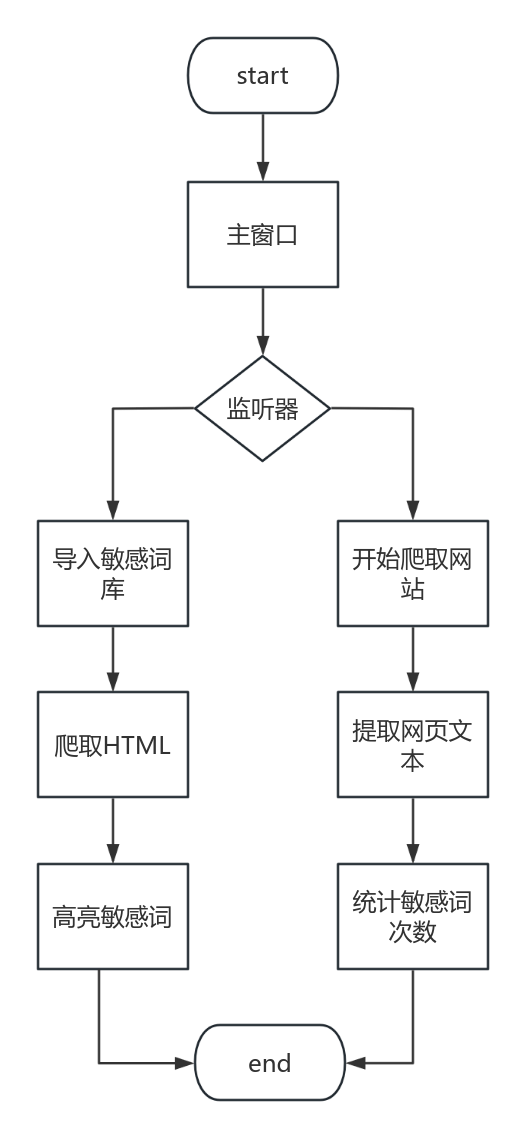
* + 主窗口提供用户输入网址的文本框和爬取按钮，以及显示爬取结果的文本区域。文本区域包括两个部分，一个显示HTML源代码，另一个显示提取后的纯文本。
  + 提供导入敏感词库和匹配敏感词的功能。用户可以导入敏感词库文件，系统会将敏感词显示在界面上，并在网页文本中查找和高亮显示敏感词。

**进度条**

* + 在爬取过程中显示进度条，提示用户当前爬取状态。进度条通过一个独立的JDialog实现，显示当前正在爬取的网址。

### 工作流程

* 用户在主窗口输入网址并点击爬取按钮。系统获取用户输入的网址，并调用URL和URLConnection类实现网页HTML代码的爬取。
* 对获取的HTML代码进行正则表达式匹配，提取纯文本内容。系统使用多种正则表达式过滤HTML标签、script标签、style标签以及多余的空格和图案乱码等，最终获取纯文本内容。
* 用户可以导入敏感词库，系统从文件中读取敏感词，并显示在界面上的敏感词区域。用户点击匹配按钮后，系统在网页文本中查找敏感词，并使用Highlighter类高亮显示匹配到的敏感词。
* 系统提供批量爬取功能，用户可以导入包含多个网址的文件，系统依次爬取每个网址，统计敏感词的出现次数，并将结果保存到文件中。



### ****技术实现****

**主窗口**

* + 使用Java Swing构建用户界面，包括输入文本框、按钮、文本区域等组件。界面布局使用BorderLayout和GridLayout等布局管理器。
  + 使用URL和URLConnection类实现网页HTML代码的爬取。通过BufferedReader逐行读取HTML源代码，并保存到文本区域中。
  + 使用正则表达式进行HTML标签过滤，提取纯文本内容。系统定义了多种正则表达式匹配符，包括匹配普通标签、script标签、style标签、连续空格或回车以及图案乱码等。

**敏感词库**

* + 提供文件导入功能，读取敏感词库文件中的词汇，并将其保存到ArrayList中。用户可以通过文件选择对话框选择敏感词库文件，系统读取文件内容并显示在界面上的敏感词区域。

**敏感词匹配**

* + 使用Highlighter类实现敏感词的高亮显示。系统在网页文本中查找敏感词，并在文本区域中高亮显示匹配到的词汇。高亮显示颜色设置为黄色，方便用户识别。

**批量爬取**

* + 使用多线程实现批量网址爬取，提高效率。每个线程爬取一个网址，并在爬取完成后统计敏感词的出现次数。
  + 系统提供进度条显示当前爬取状态。通过JDialog实现进度条界面，提示用户当前正在爬取的网址。
  + 将爬取结果和敏感词统计数据保存到文件中。系统在批量爬取完成后，将每个网址的敏感词统计结果以及总的统计结果保存到文本文件中，方便用户查看和分析。

# 主要模块的算法说明

## 实验一

### 服务器端 (Server)

#### 类和变量

* `server 类继承自 `JFrame`，并实现了 `Runnable`、`ListSelectionListener` 和 `ActionListener` 接口。
* 服务器使用一个 `ServerSocket` 对象 `ss` 监听端口9999。
* `users` 是一个 `ArrayList`，用于存储所有连接的客户端线程 `ChatThread`。
* 图形界面包括用户列表 `JList<String> userList` 和两个按钮 `jbt`（踢出聊天室）和 `jbt1`（群发消息）。

#### 构造方法

* 初始化服务器窗口，包括设置标题、布局、尺寸等。
* 创建一个 `ServerSocket` 对象，监听端口9999。
* 启动一个新线程，调用 `run` 方法，开始监听客户端连接。

#### run方法

* 服务器端持续运行，接受客户端连接，每当有客户端连接时，创建一个新的 `ChatThread` 对象，并将其加入 `users` 列表。

#### ChatThread类

* 继承自 `Thread`，每个客户端连接都会创建一个 `ChatThread` 对象。
* 维护客户端的 `Socket`、输入输出流、昵称等信息。
* 在 `run` 方法中，持续读取客户端发送的消息，并根据消息类型进行处理，如上线、消息转发、下线等。

#### actionPerformed方法

* 实现按钮点击事件的处理，包括群发消息和踢出用户。
* 群发消息：读取输入框内容，通过 `sendMessage` 方法发送给所有用户。
* 踢出用户：发送下线消息给指定用户，并从用户列表中移除。

### 客户端 (Client)

#### 类和变量

* client` 类继承自 `JFrame`，并实现了 `Runnable` 和 `ActionListener` 接口。
* 图形界面包括菜单栏、好友列表、消息显示区域、消息输入区域等。
* 维护与服务器通信的 `Socket`、输入输出流、昵称等信息。

#### 构造方法

* 初始化客户端窗口，包括设置标题、布局、尺寸等。
* 弹出输入框获取用户昵称。
* 创建与服务器的连接，并启动一个新线程，调用 `run` 方法，开始监听服务器发送的消息。

#### `run` 方法

* 客户端持续运行，读取服务器发送的消息，并根据消息类型进行处理，如上线通知、消息显示、好友列表更新、下线通知等。

#### `actionPerformed` 方法

* 实现按钮点击事件的处理，包括发送消息、私聊、群聊等。
* 发送消息：读取输入框内容，通过 `ps.println` 方法发送给服务器。
* 私聊：创建私聊窗口，并实现私聊消息发送功能。
* 群聊：创建群聊窗口，并实现群聊消息发送功能。

### 主要功能

1. 用户登录/下线：客户端启动后，输入昵称并发送给服务器，服务器接收并更新所有在线用户列表。用户下线时，服务器通知其他用户。

2. 消息发送：包括群发消息和私聊消息，服务器接收消息后，根据消息类型进行转发。

3. 用户管理：服务器端可以踢出用户，客户端实时更新用户列表。

4. 图形界面：使用 Swing 实现图形界面，提供良好的用户体验。

### 代码结构和设计

* 面向对象设计：服务器和客户端分别封装成独立的类，客户端通过线程处理与服务器的通信。
* 多线程：服务器使用多线程处理多个客户端连接，每个客户端连接在独立的线程中运行，保证并发处理能力。
* 事件驱动：通过实现 `ActionListener` 和 `ListSelectionListener` 接口，处理用户界面操作。

## 实验二

### 功能模块结构

#### ****界面模块****

* 创建主窗口及其布局。
* 初始化Swing组件。
* 设置事件监听器。

#### ****网页爬取模块****

* 使用URL类连接网页。
* 通过URLConnection和BufferedReader爬取HTML代码。
* 使用正则表达式提取网页文本。

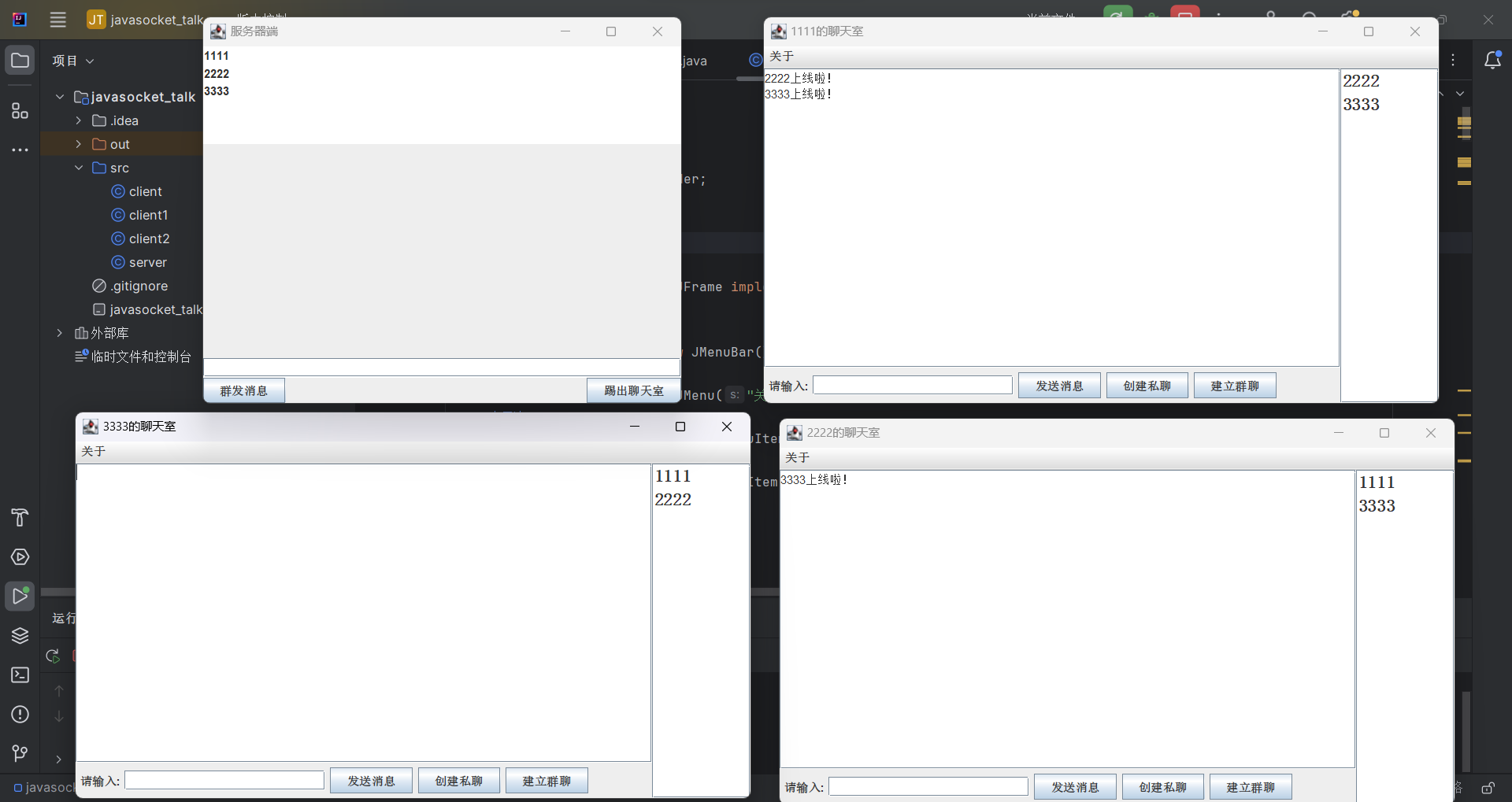
#### 敏感词处理模块

* 导入敏感词库。
* 高亮显示匹配的敏感词。
* 爬取多个网址，并统计敏感词出现次数。

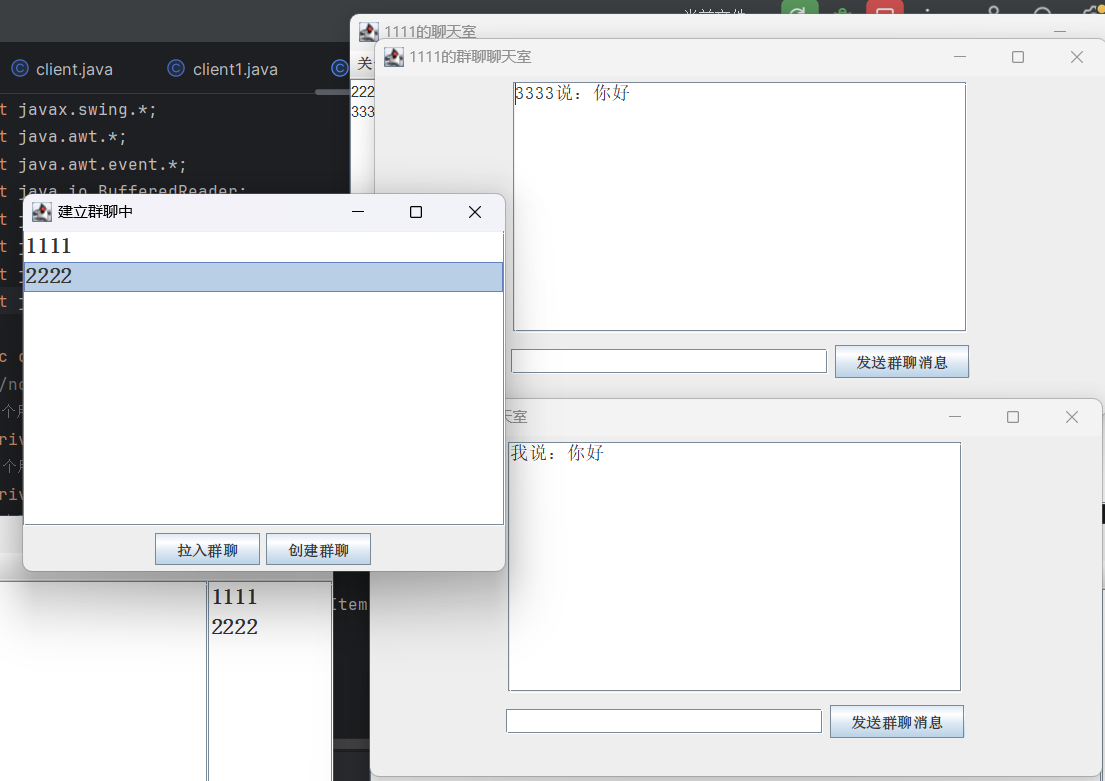
# 运行结果

## 实验一

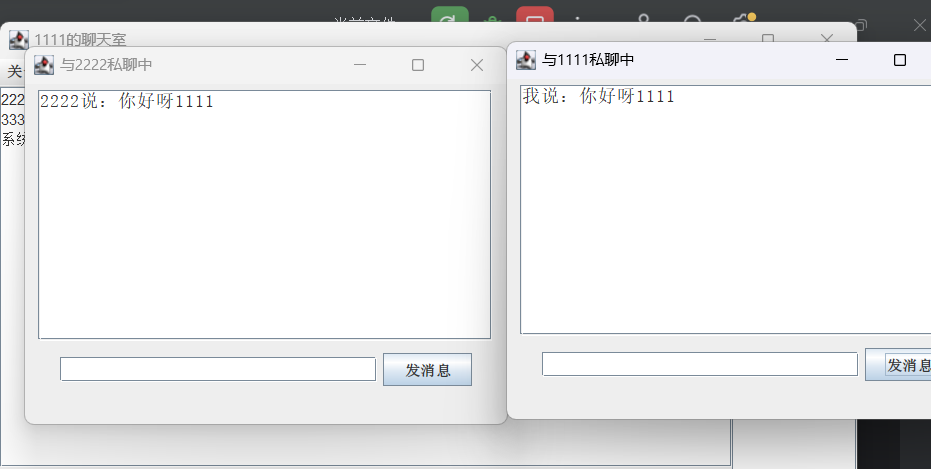
### 主界面，包括上下线刷新



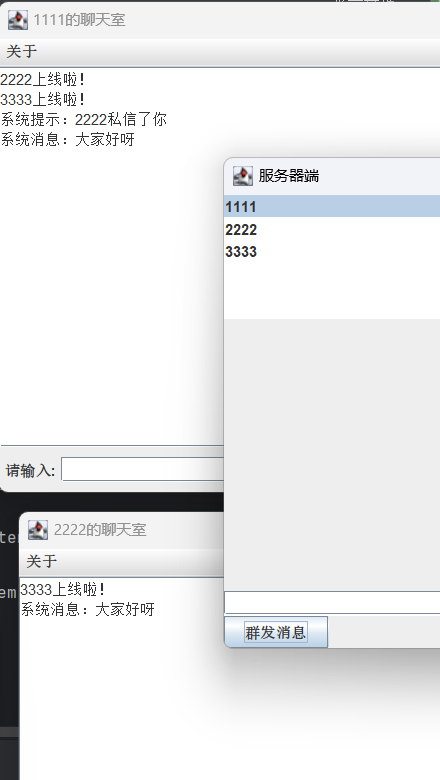
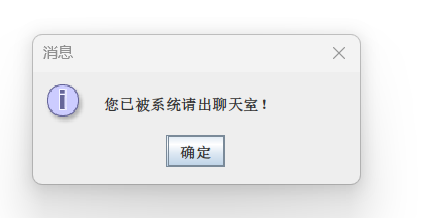
### 实现和其他用户的群聊，建立小群与解散功能



### 实现私聊



### 服务器端实现群发消息与用户强制下线

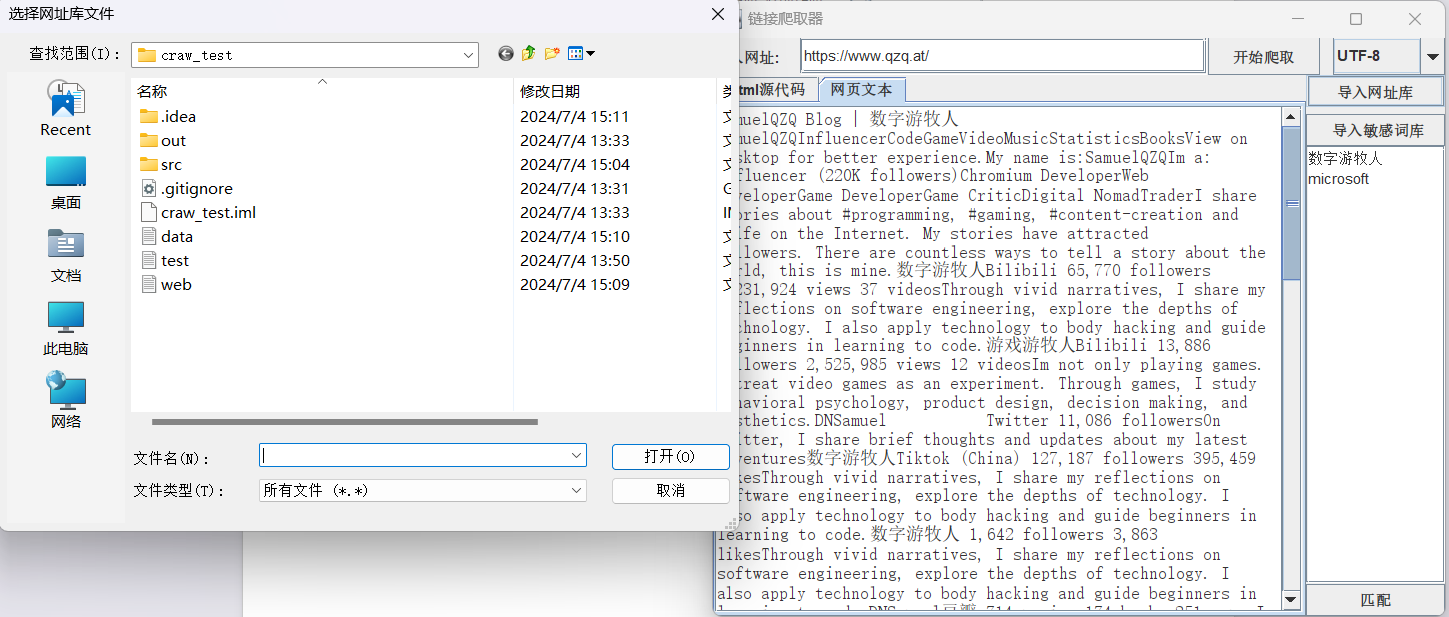
 

## 实验二

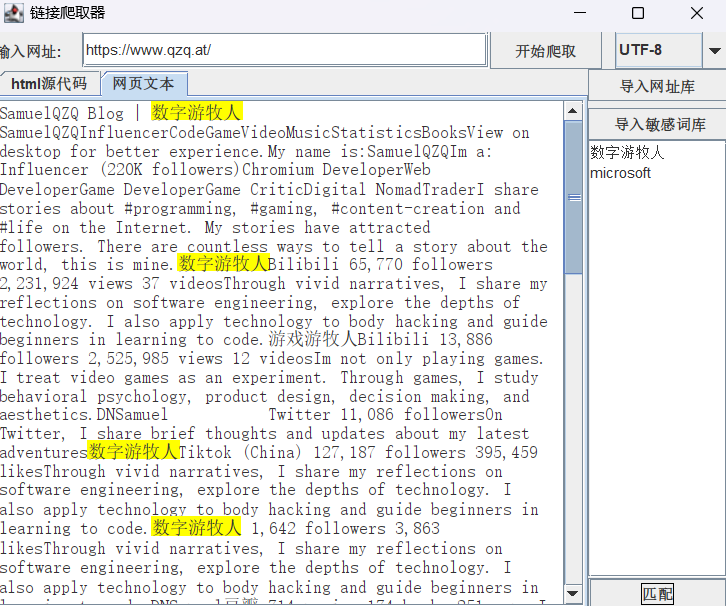
### 主界面

## 

### 导入敏感库和网址库



### 统计敏感词并高亮敏感词





# 课程设计总结

## 课程设计中的问题、解决过程及体会

### 1. Socket编程中的通信问题

问题描述：在实现客户端和服务器之间的通信时，初期遇到了连接不稳定、数据传输不完整的问题。

解决过程：

1. 调试和日志记录：通过在客户端和服务器端添加日志记录，详细记录每次连接和数据传输的过程，以便找出问题所在。

2. 连接管理：改进了连接管理机制，确保每次连接建立前先检查已有连接状态，避免重复连接导致的冲突。

3. 数据完整性检查：增加了数据传输的校验机制，确保每次传输的数据完整性，通过确认接收方成功接收数据后再关闭连接。

体会：通过这个过程，我们深入了解了Socket编程的细节，特别是如何处理网络通信中的异常情况，提高了代码的健壮性。

### 2. 正则表达式匹配的准确性

问题描述：在提取网页纯文本时，使用的正则表达式有时会匹配到不需要的内容或遗漏重要内容。

解决过程：

1. 优化正则表达式：逐步优化正则表达式，分别处理不同类型的HTML标签，如普通标签、script标签、style标签等。

2. 多次过滤：采用多次过滤的策略，先进行粗过滤，再进行细过滤，确保最大限度提取有效文本。

3. 测试与调整：通过大量测试网页，检查提取效果，并根据测试结果不断调整正则表达式，直到达到满意的效果。

体会：学习了正则表达式的高级用法，以及如何在实际应用中不断调试和优化，提高了对文本处理的理解和能力。

### 3. 敏感词匹配的性能问题

问题描述：在匹配敏感词时，文本较长和敏感词库较大时，性能下降明显，影响用户体验。

解决过程：

1. 数据结构优化：将敏感词存储在HashSet中，提高匹配效率。

2. 多线程处理：使用多线程处理文本分段，分别匹配敏感词，提高整体处理速度。

3. 算法优化：研究并实现了更高效的字符串匹配算法，如Aho-Corasick算法，用于多模式匹配。

体会：深刻理解了数据结构和算法在实际应用中的重要性，通过优化提高了程序的性能和响应速度。

## 课程设计的收获

1. 理论与实践的结合：将所学的Java编程、网络编程、正则表达式等知识应用到实际项目中，加深了对理论知识的理解和掌握。

2. 问题解决能力：在实际开发过程中遇到各种问题，通过不断调试、学习和优化，提高了我们分析问题和解决问题的能力。

3. 团队合作精神：通过团队合作，大家分工明确，互相配合，顺利完成了项目，体会到了团队协作的重要性。

4. 代码质量意识：学会了编写规范、高效、易维护的代码，注重代码的可读性和可扩展性。

## 对程序开发的认识与思考

1. 编程不仅仅是写代码：编程不仅是实现功能，更重要的是解决实际问题。需要综合考虑用户需求、性能优化、代码维护等多个方面。

2. 学习与积累：技术更新迭代很快，需要不断学习新知识，积累经验。通过实践不断提升自己的编程能力和技术水平。

3. 重视设计与规划：一个好的程序需要良好的设计和规划。提前做好系统设计和架构规划，可以减少后期的修改和维护成本。

4. 调试与优化：在开发过程中，调试和优化是必不可少的步骤。通过不断调试和优化，可以提高程序的稳定性和性能，提升用户体验。

这次课程设计让我们深入理解了编程的过程和方法，提高了我们的实践能力和综合素质，为以后的学习和工作打下了坚实的基础。

# 参考文献

1. 《JAVA语言程序设计》 清华大学出版社 郭克华著

# 附录

## 实验一

### Server

import java.awt.BorderLayout;  
import java.awt.Dimension;  
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;  
  
import javax.swing.\*;  
import javax.swing.event.ListSelectionEvent;  
import javax.swing.event.ListSelectionListener;  
public class server extends JFrame implements Runnable, ListSelectionListener, ActionListener{  
 private Socket s = null;  
 private ServerSocket ss = null;  
 private ArrayList<ChatThread> users = new ArrayList<ChatThread>();  
 DefaultListModel<String> dl = new DefaultListModel<String>();  
 private 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。  
 private JPanel jpl = new JPanel();  
 private JButton jbt = new JButton("踢出聊天室");  
 private JButton jbt1 = new JButton("群发消息");  
 //群发消息输入栏  
 private JTextField jtf = new JTextField();  
 public server() throws Exception{  
 this.setTitle("服务器端");  
 this.add(userList, "North");//放在东面  
 this.add(jpl, "South");  
 jtf.setColumns(1);  
 userList.setPreferredSize(new Dimension(200,100));  
 jpl.setLayout(new BorderLayout());  
 jpl.add(jtf, BorderLayout.NORTH);  
 jpl.add(jbt,BorderLayout.EAST);//踢出聊天室  
 jpl.add(jbt1, BorderLayout.WEST);//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(500, 400);  
 this.setVisible(true);  
 this.setAlwaysOnTop(true);  
 ss = new ServerSocket(9999);  
 new Thread(this).start();//监听用户端的加入  
 }  
 public void run () {  
 while(true){  
 try{  
 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  
 System.out.println("用户" + ct);  
 //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();  
 javax.swing.JOptionPane.showMessageDialog(this,"服务器异常！");  
 System.exit(0);  
 }  
 }  
 }  
 public class ChatThread extends Thread{  
 Socket s = null;  
 private BufferedReader br = null;  
 private PrintStream ps = null;  
 public boolean canRun = true;  
 String nickName = null;  
 ArrayList<String> ul = new ArrayList<>();  
 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(true) {  
 try {  
 String msg = br.readLine();  
 System.out.println(msg);  
 String[] strs = msg.split("#");  
 // 继续处理消息  
  
  
 if(strs[0].equals("LOGIN")){//收到来自客户端的上线消息  
 nickName = strs[1];  
 dl.addElement(nickName);  
 userList.repaint();  
 sendMessage(msg);  
 System.out.println("用户" + nickName);  
 }else if(strs[0].equals("GUSERS")){//USER消息，为新建立的客户端更新好友列表  
 ul.add(strs[1]);  
 }else if(strs[0].equals("MSG") ){  
 sendMessage(msg);  
 }else if(strs[0].equals("SMSG") || strs[0].equals("FSMSG")){  
  
 sendMessageToUser(strs[2],msg);  
 }else if(strs[0].equals("GMSG")){  
 sendMessageToUser(strs[2],msg);  
 }else if(strs[0].equals("GMSGOVER")){  
 sendMessageToUser(strs[2],msg);  
 ul.clear();  
 }else if(strs[0].equals("FGMSG")){  
 String ss = null;  
 System.out.println("fashong"+ul+nickName);  
 for (String element : ul) {  
 ss = "GUSERS#" + element;  
 sendMessageToUser(strs[2],ss);  
 System.out.println("fashong"+ul);  
 }  
 sendMessageToUser(strs[2],msg);  
  
 }else if(strs[0].equals("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();  
 }  
 }catch (Exception ex){  
// System.out.println("66");  
 }  
 }  
  
 }  
  
  
  
 }  
 public void actionPerformed(ActionEvent e) {  
 String label = e.getActionCommand();  
 if(label.equals("群发消息")){  
 handleAll();  
 }else if(label.equals("踢出聊天室")){  
 try {  
 handleExpel();  
 } catch (IOException e1) {  
 e1.printStackTrace();  
 }  
 }  
 }  
 public void handleAll(){  
 if(!jtf.getText().equals("")){  
 sendMessage("ALL#" + jtf.getText());  
 //发送完后，是输入框中内容为空  
 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  
  
 public void sendMessage(String msg){ //服务器端发送给所有用户  
 for(ChatThread ct : users){  
 ct.ps.println(msg);  
 }  
 }  
 public void sendMessageToUser(String userID, String msg) {  
 // 遍历所有用户，查找指定用户  
 for (ChatThread ct : users) {  
 // 假设ct.userID是每个用户的唯一标识  
 if (ct.nickName.equals(userID)) {  
 // 找到用户，发送消息  
 ct.ps.println(msg);  
 return; // 发送成功后退出循环  
 }  
 }  
 // 如果没有找到用户，可以添加适当的错误处理逻辑  
 System.out.println("用户不存在或未找到： " + userID);  
 }  
 @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

import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.\*;  
import java.io.BufferedReader;  
import java.io.InputStreamReader;  
import java.io.PrintStream;  
import java.net.Socket;  
import java.util.ArrayList;  
  
public class client extends JFrame implements Runnable,ActionListener {  
 //north  
 private JMenuBar bar = new JMenuBar();  
 private JMenu menu = new JMenu("关于");  
 private JMenuItem about = new JMenuItem("关于本软件");  
 private JMenuItem exit = new JMenuItem("退出");  
 JPanel north = new JPanel();  
 private JPanel jpl = new JPanel();  
 //west  
 JPanel west = new JPanel();  
 DefaultListModel<String> dl = new DefaultListModel<String>();//用来修改JList  
 ArrayList<String> ul = new ArrayList<>();  
 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(20);  
  
 JButton jButton = new JButton("发消息");  
 JButton jButton1 = new JButton("发送群聊消息");  
 private JButton jbt = new JButton("发送消息");  
 private JButton jbt1 = new JButton("创建私聊");  
 private JButton jbt2 = new JButton("建立群聊");  
 private JButton jbt3 = new JButton("拉入群聊");  
 private JButton jbt4 = new JButton("创建群聊");  
 private BufferedReader br = null;  
 private PrintStream ps = null;  
 private String nickName = null;  
  
 //私聊面板  
 JTextArea jTextArea = new JTextArea(11,45);  
 JScrollPane js1 = new JScrollPane(jTextArea);  
 JTextField jTextField = new JTextField(25);  
 String suser = new String();  
 String ssuser = new String();  
 double MAIN\_FRAME\_LOC\_X;//父窗口x坐标  
 double MAIN\_FRAME\_LOC\_Y;//父窗口y坐标  
 JTextArea gTextArea = new JTextArea(11,45);  
 JScrollPane js2 = new JScrollPane(gTextArea);  
 JTextField gTextField = new JTextField(25);  
  
 boolean FirstSecret = true;//是否第一次私聊  
 boolean FirstSecretgroup = 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{  
 //north 菜单栏  
 bar.add(menu);  
 menu.add(about);  
 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("隶书",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);  
 gTextArea.setEditable(false);  
  
 BorderLayout bl3 = new BorderLayout();  
 center.setLayout(bl3);  
 FlowLayout fl = new FlowLayout(FlowLayout.LEFT);  
 operPane.setLayout(fl);  
 operPane.add(input);  
 operPane.add(jtf);  
 operPane.add(jbt);  
 operPane.add(jbt1);  
 operPane.add(jbt2);  
 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);  
 jbt2.addActionListener(this);  
 jbt3.addActionListener(this);  
 jbt4.addActionListener(this);  
 this.setDefaultCloseOperation(EXIT\_ON\_CLOSE);  
 //this.setAlwaysOnTop(true);  
  
 nickName = JOptionPane.showInputDialog("用户名：");  
 this.setTitle(nickName + "的聊天室");  
 this.setSize(700,400);  
 this.setVisible(true);  
  
 Socket s = new Socket("127.0.0.1", 9999);  
 br = new BufferedReader(new InputStreamReader(s.getInputStream()));  
 ps = new PrintStream(s.getOutputStream());  
 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("#");  
 // 继续处理消息  
  
  
 System.out.println(msg+nickName);  
 //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 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("GUSERS")){//USER消息，为新建立的客户端更新好友列表  
 ul.add(strs[1]);  
 } 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  
 javax.swing.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 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 if(strs[0].equals("GMSG")){  
  
 gTextArea.append(strs[1] + "说：" + strs[3] + "\n");  
 jta.append("系统提示：" + strs[1] + "群聊了你" + "\n");  
 }else if(strs[0].equals("GMSGOVER")){  
 gTextArea.append("群聊已结束，请关闭窗口\n");  
 ul.clear();  
 FirstSecretgroup = true;  
  
 }else if(strs[0].equals("FGMSG"))  
 {  
 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)) {  
 FirstSecretgroup = false;  
 gTextArea.append(strs[1] + "说：" + strs[3] + "\n");  
// jta.append("系统提示：" + strs[1] + "私信了你" + "\n");  
 handlgroupS();  
  
 }  
 else {  
 gTextArea.append("我说：" + strs[3] + "\n");  
// System.out.println("668");  
 }  
 }  
  
 }catch (Exception ex){//如果服务器端出现问题，则客户端强制下线  
 javax.swing.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();  
 System.out.println(label);  
 if(label.equals("发送消息")){//群发  
 handleSend();  
 }else if(label.equals("创建私聊") && !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("发送群聊消息")){  
 grouphandleSSS();//群发消息  
// jButton1.removeActionListener(this);  
 }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\n\n3.可以进行群聊\n\n");  
 }else if(label.equals("退出")){  
 JOptionPane.showMessageDialog(this, "您已成功退出！");  
 ps.println("OFFLINE#" + nickName);  
 System.exit(0);  
 }else if(label.equals("建立群聊")){  
 handlgroupSS();  
 }else if(label.equals("拉入群聊")){  
 java.util.List<String> selectedUsers = userList.getSelectedValuesList();  
  
 if(selectedUsers.isEmpty()){  
 // 如果没有选择用户，显示提示信息  
 JOptionPane.showMessageDialog(this, "未选择用户！请选择至少一个用户再加入群聊。");  
 }  
 else{  
 ssuser = userList.getSelectedValuesList().get(0);//获得被选择的用户  
 if(!ul.contains(ssuser)) {  
 // 如果用户不存在，添加到列表中  
 ul.add(ssuser);  
 System.out.println("Selected Users: " + ul);  
 }else {  
 System.out.println("666 " );  
 }  
  
 }  
 //System.out.println("Selected Users: " + ul);  
 }else if(label.equals("创建群聊")){  
 handlgroupS();//创建群聊窗口else{  
 }  
 }  
  
 public void handleSS(){//在私聊窗口中发消息  
 String name=sender;  
 String msg = new String();  
 if(sender.equals(nickName)) {  
 name = receiver;  
 }  
 if(FirstSecret) {  
 msg = jTextField.getText();  
 if (!msg.isEmpty()) {  
 ps.println("FSMSG#" + nickName + "#" + name + "#" + msg);  
 jTextArea.append("我说："+msg+ "\n");  
 jTextField.setText("");  
 FirstSecret = false;  
 }else {  
 jTextArea.append("不能发送空消息"+ "\n");  
 }  
 }  
 else {  
 msg = jTextField.getText();  
 if (!msg.isEmpty()) {  
 ps.println("SMSG#" + nickName + "#" + name + "#" + msg);  
 jTextArea.append("我说："+msg+ "\n");  
 jTextField.setText("");  
 }else {  
 jTextArea.append("不能发送空消息"+ "\n");  
 }  
 }  
 }  
 public void grouphandleSSS() {// 在群聊窗口中发消息  
 String msg = new String();  
 if (ul.contains(nickName)) {  
 // 如果不存在，则添加  
 ul.remove(nickName);  
 }  
  
 System.out.println(ul);  
 if (FirstSecretgroup) {  
 msg = gTextField.getText();  
 if (!msg.isEmpty()) {  
 for (String element : ul) {  
 ps.println("GUSERS#" + element);  
 System.out.println(msg);  
 System.out.println("1");  
 }  
 ps.println("GUSERS#" + nickName);  
 for (String name : ul) {  
  
 ps.println("FGMSG#" + nickName + "#" + name + "#" + msg);  
 System.out.println(msg);  
 }  
  
 gTextArea.append("我说："+msg+ "\n");  
 FirstSecretgroup = false;  
 }else {  
 gTextArea.append("消息不能为空"+"\n");  
 }  
 } else {  
 msg = gTextField.getText();  
 if (!msg.isEmpty()) {  
 for (String name : ul) {  
 ps.println("GMSG#" + nickName + "#" + name + "#" + msg);  
 System.out.println(msg);  
 }  
 gTextArea.append("我说："+msg+ "\n");  
 }else {  
 gTextArea.append("消息不能为空"+"\n");  
 }  
 }  
// System.out.println("666");  
 gTextField.setText("");  
// System.out.println("667");  
 }  
 public void handleSend(){//大群发消息  
 //发送信息时标识一下来源  
 String msg =jtf.getText();  
 if (!msg.isEmpty()) {  
 ps.println("MSG#" + nickName + "#" + msg);  
 jta.append("我说："+msg+ "\n");  
 //发送完后，是输入框中内容为空  
 jtf.setText("");  
 }else {  
 jta.append("消息不能为空"+ "\n");  
 }  
 }  
 public void handleSec(String name){ //建立私聊窗口  
 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 + "私聊中");  
 jFrame.setVisible(true);  
  
 jTextField.setFocusable(true);//设置焦点  
  
 jFrame.addWindowListener(new WindowAdapter() {  
 @Override  
 public void windowClosing(WindowEvent e) {  
 jTextArea.setText("");  
 jButton.removeActionListener(client.this);  
 FirstSecret = true;  
 }  
 });  
 }//私聊窗口  
 public void handlgroupSS(){ //群聊一阶窗口  
 JFrame frame = new JFrame();//新建了一个窗口  
 frame.add(listPane, BorderLayout.CENTER);  
 jpl.add(jbt3);  
 jpl.add(jbt4);  
 frame.add(jpl, BorderLayout.SOUTH);  
 frame.setSize(400,310);  
 frame.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  
 frame.setTitle("建立群聊中");  
 frame.setVisible(true);  
  
  
 }  
 public void handlgroupS(){ //建立群聊窗口  
 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(gTextField);  
 JPL.add(jButton1);  
 JPL2.add(js2,BorderLayout.CENTER);  
 JPL2.add(JPL,BorderLayout.SOUTH);  
 jFrame.add(JPL2);  
  
 jButton1.addActionListener(this);  
 gTextArea.setFont(new Font("宋体", Font.PLAIN,15));  
 jFrame.setSize(600,310);  
 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(nickName+"的群聊聊天室");  
 jFrame.setVisible(true);  
  
 gTextField.setFocusable(true);//设置焦点  
  
 jFrame.addWindowListener(new WindowAdapter() {  
 @Override  
 public void windowClosing(WindowEvent e) {  
 for (String name : ul) {  
  
 ps.println("GMSGOVER#" + nickName + "#" + name );  
 }  
 gTextArea.setText("");  
 ul.clear();  
 jButton1.removeActionListener(client.this);  
 System.out.println("66");  
 FirstSecretgroup = true;  
 }  
 });  
 }  
 public static void main(String[] args)throws Exception{  
 new client();  
 }  
}

## 实验二

import java.awt.FileDialog;  
import java.awt.BorderLayout;  
import java.awt.Color;  
import java.awt.Dimension;  
import java.awt.Font;  
import java.awt.GridLayout;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.io.BufferedReader;  
import java.io.File;  
import java.io.FileNotFoundException;  
import java.io.FileReader;  
import java.io.IOException;  
import java.io.InputStreamReader;  
import java.io.PrintStream;  
import java.net.URL;  
import java.net.URLConnection;  
import java.util.ArrayList;  
import java.util.regex.Matcher;  
import java.util.regex.Pattern;  
import javax.swing.JButton;  
import javax.swing.JComboBox;  
import javax.swing.JDialog;  
import javax.swing.JFileChooser;  
import javax.swing.JFrame;  
import javax.swing.JLabel;  
import javax.swing.JOptionPane;  
import javax.swing.JPanel;  
import javax.swing.JProgressBar;  
import javax.swing.JScrollPane;  
import javax.swing.JTabbedPane;  
import javax.swing.JTextArea;  
import javax.swing.JTextField;  
import javax.swing.UIManager;  
import javax.swing.UnsupportedLookAndFeelException;  
import javax.swing.text.BadLocationException;  
import javax.swing.text.DefaultHighlighter.DefaultHighlightPainter;  
import javax.swing.text.Highlighter;  
  
//采用JavaURL类连接网页网页。  
//爬取html代码：使用URLConection配合BufferedReader逐行读取html源代码。  
//提取网页文本：使用Java正则表达式匹配html源代码，过滤其中的标签等信息。  
//高亮显示敏感词：获取文本框中的Highlighter进行高亮显示，匹配到敏感词则进行高亮显示  
//爬取多个网址：分别爬取每个网址，统计每个网址中的敏感词数据，最后进行数据汇总，并将数据都写入文本中。  
  
public class craw extends JFrame implements ActionListener{  
 //创建主窗口  
 private JPanel jPanel=new JPanel();  
 private JPanel jpl1=new JPanel();  
 private JPanel jpl2=new JPanel();  
 private JPanel jpl3=new JPanel();  
 private JPanel jpl4=new JPanel();  
 private JPanel jpl5=new JPanel();  
 private JPanel jpl6=new JPanel();  
 private JPanel jpl7=new JPanel();  
 private JPanel jpl8=new JPanel();  
 //初始化Swing的组件  
 private JLabel siteWarn=new JLabel("输入网址:");  
 private JTextField siteField=new JTextField(25);  
 private JScrollPane siteSPane=new JScrollPane(siteField);  
 private JButton goSpider=new JButton("开始爬取");  
 private JTextArea htmlArea=new JTextArea(15, 25);  
 private JScrollPane htmlSPane=new JScrollPane(htmlArea);  
 private JTextArea textArea=new JTextArea(15,25);  
 private JScrollPane textSPane=new JScrollPane(textArea);  
 private JTabbedPane tabPane=new JTabbedPane();  
 private JTextArea sensWord=new JTextArea(8,25);  
 private JScrollPane wordPane=new JScrollPane(sensWord);  
 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 String textType="UTF-8";  
  
 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  
 //通过正则表达式将爬取的文本进行过滤，除去标签信息只保留文本信息  
 //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 pattern3=Pattern.compile(regExHtml, Pattern.CASE\_INSENSITIVE);  
 private Pattern pattern1=Pattern.compile(regExScript,Pattern.CASE\_INSENSITIVE);  
 private Pattern pattern2=Pattern.compile(regExStyle,Pattern.CASE\_INSENSITIVE);  
 private Pattern pattern4=Pattern.compile(regExSpace, Pattern.CASE\_INSENSITIVE);  
 private Pattern pattern5=Pattern.compile(regExImg,Pattern.CASE\_INSENSITIVE);  
  
 public craw() throws IOException {  
 //设置界面风格  
 try {  
 UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());  
 } catch (ClassNotFoundException e1) {  
 // TODO Auto-generated catch block  
 e1.printStackTrace();  
 } catch (InstantiationException e1) {  
 // TODO Auto-generated catch block  
 e1.printStackTrace();  
 } catch (IllegalAccessException e1) {  
 // TODO Auto-generated catch block  
 e1.printStackTrace();  
 } catch (UnsupportedLookAndFeelException e1) {  
 // TODO Auto-generated catch block  
 e1.printStackTrace();  
 }  
  
 this.setTitle("链接爬取器");  
 this.setLocation(400, 200);  
 this.setSize(600, 500);  
 this.setDefaultCloseOperation(EXIT\_ON\_CLOSE);  
 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  
 jpl1.setLayout(new BorderLayout());  
 siteWarn.setPreferredSize(new Dimension(70,30));  
 siteSPane.setPreferredSize(new Dimension(300, 30));  
 goSpider.setPreferredSize(new Dimension(90, 30));  
 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));  
 jpl2.setLayout(new BorderLayout());  
 jpl2.add(htmlSPane,BorderLayout.CENTER);  
 //设置布局  
 jpl8.setLayout(new GridLayout(2, 1, 10,5));  
 jpl8.add(siteLib);  
 jpl8.add(openLib);  
  
 jpl3.setLayout(new BorderLayout());  
 sensWord.setLineWrap(true);  
 sensWord.setEditable(false);  
 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);  
 jpl4.setLayout(new BorderLayout());  
 jpl4.add(textSPane,BorderLayout.CENTER);  
  
 tabPane.add("html源代码", jpl2);  
 tabPane.add("网页文本",jpl4);  
 jpl7.setLayout(new BorderLayout());  
 jpl7.add(tabPane,BorderLayout.CENTER);  
  
 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) {//要爬取的网址  
  
 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 urlConne=url.openConnection(); //连接  
 urlConne.connect();  
 //获取输入流  
 BufferedReader br=new BufferedReader(new InputStreamReader(urlConne.getInputStream(),textType));  
 //getInputStream（输入流）->InputStreamReader（字符流）->BufferedReader（包装）  
 System.out.println("开始爬取");  
 while(true) { //爬取到结束  
 str=br.readLine();  
 //一行一行爬取  
 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(""); //匹配普通标签  
 matcher=pattern2.matcher(str);  
 str=matcher.replaceAll(""); //匹配script标签  
 matcher=pattern3.matcher(str);  
 str=matcher.replaceAll(""); //匹配style标签  
 matcher=pattern4.matcher(str);  
 str=matcher.replaceAll("\n"); //匹配连续回车或空格  
 matcher=pattern5.matcher(str);  
 str=matcher.replaceAll(""); //匹配网页图案出现的乱码  
 return str; //返回文本  
 }  
  
 //从文件中读取敏感词  
 public void getLib() {  
 FileDialog fd = new FileDialog(this, "选择敏感词库文件", FileDialog.LOAD);  
 fd.setVisible(true);  
 String filename = fd.getFile();  
 if (filename == null) {  
 return; // 用户取消选择  
 }  
 wordList.clear(); // 清空之前的记录  
 sensWord.setText("");  
 File choosenLib = new File(fd.getDirectory(), filename); // 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));  
 String str;  
 while ((str = br.readLine()) != null) {  
 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) {  
 JOptionPane.showMessageDialog(null, "文件不存在");  
 e1.printStackTrace();  
 } catch (IOException e1) {  
 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;  
 }  
  
 FileDialog fd = new FileDialog(this, "选择网址库文件", FileDialog.LOAD);  
 fd.setVisible(true);  
 String filename = fd.getFile();  
 if (filename == null) {  
 return; // 用户取消选择  
 }  
  
 File file = new File(fd.getDirectory(), filename);  
 new SpiderAll(this, file).start(); // 开启线程爬取  
 }  
  
  
 public static void main(String[] args) throws IOException {  
 new craw();  
 }  
  
 @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  
 private MyProgressBar mpb=null; //进度条  
 //构造函数初始化  
 public SpiderOne(JFrame fa,String s) {  
 website=s;  
 mpb=new MyProgressBar(fa, "Spiding");  
 }  
  
 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("");  
 mpb.setText("爬取"+website+"中..."); //设置进度条界面标题  
 mpb.setVisible(true); //显示进度条  
 String html=getHtml(website); //开始爬取  
 mpb.dispose(); //关闭进度条  
 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  
 private MyProgressBar mpb=null; //进度条  
 //构造函数初始化  
 public SpiderAll(JFrame fa,File f) {  
 file=f;  
 mpb=new MyProgressBar(fa, "Spiding");  
 }  
  
 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();  
 mpb.setVisible(true); //显示进度条  
 while(true) {  
 String website=brr.readLine();  
 if(website==null) break;  
 mpb.setText("爬取"+website+"中..."); //设置进度条界面标题  
 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");  
 }finally {  
 mpb.dispose(); //关闭进度条  
 }  
 }  
 }  
}  
  
//进度条设计  
class MyProgressBar extends JDialog {  
  
 private static final long serialVersionUID = 1L;  
 private JPanel jPanel=new JPanel();  
 private JProgressBar jpb=new JProgressBar(); //进度条  
 private JLabel curSpiding=new JLabel(); //显示当前网址  
 //构造函数初始化,设置父窗口以及标题  
 public MyProgressBar(JFrame f,String title) {  
 super(f,title);  
 this.setLocation(f.getWidth()/2+(int)f.getLocation().getX()/2-80, f.getHeight()/2+(int)f.getLocation().getY()/2-30);  
 this.setSize(400,100);  
 this.setDefaultCloseOperation(DISPOSE\_ON\_CLOSE);  
  
 jpb.setString("努力加载中...");  
 jpb.setIndeterminate(true); //设置进度条为不确定模式  
 jpb.setStringPainted(true);  
 jpb.setBorderPainted(false);  
 jpb.setForeground(Color.RED); //设置进度条颜色  
 jpb.setBackground(Color.WHITE); //设置背景  
 curSpiding.setPreferredSize(new Dimension(400, 30));  
  
 //界面布局  
 jPanel.setLayout(new BorderLayout());  
 jPanel.add(curSpiding,BorderLayout.NORTH);  
 jPanel.add(jpb,BorderLayout.CENTER);  
 this.add(jPanel);  
 }  
  
 //设置当前爬取网址接口  
 public void setText(String text) {curSpiding.setText(text);}  
}