

浙江大学

Java 应用技术
大 程 序 报 告



姓名： 林炬乙 学号： 3180103721 电话： 13732201652

指导老师： 楼学庆

2020~2021 秋冬学期 2021 年 1 月 17 日

目 录

| | |
|----------------------------------|-----------|
| 1 大程序简介..... | 3 |
| 1.1 选题背景及意义..... | 3 |
| 1.2 程序设计目的 | 3 |
| 2 程序开发设计..... | 4 |
| 2.1 程序设计原理 | 4 |
| 2.1.1 项目所使用到的模式..... | 4 |
| 2.1.2 server 和 client 的登录协议..... | 4 |
| 2.1.3 server 和 client 的注册协议..... | 5 |
| 2.2 程序流程..... | 6 |
| 2.3 文件列表..... | 8 |
| 2.3.1 文件函数结构..... | 8 |
| 2.3.2 多文件构成机制..... | 10 |
| 3 程序部署运行和使用说明..... | 12 |
| 3.1 编译安装..... | 12 |
| 3.2 实例运行测试 | 13 |
| 3.3 结果说明分析 | 16 |
| 3.3.1 初始界面..... | 16 |
| 3.3.2 在线功能..... | 17 |
| 3.3.3 发送信息功能..... | 18 |
| 4 难点、要点、得意点 | 19 |
| 4.1 难点..... | 19 |
| 4.1.1 多线程..... | 19 |
| 4.1.2 事件处理..... | 20 |
| 4.1.3 服务器和客户端保持连接..... | 20 |
| 4.1.4 发送给其他客户端..... | 21 |
| 4.2 编码规范..... | 22 |
| 4.3 要点..... | 24 |
| 4.3.1 实现了多个客户端连接..... | 24 |
| 4.3.2 实现了和数据库的连接..... | 25 |
| 4.4 得意点..... | 26 |
| 4.4.1 图标的设置..... | 26 |
| 4.4.2 异常的处理..... | 26 |
| 4.4.3 前后端分离..... | 26 |
| 5 结论展望 | 26 |
| 5.1 心得体会 | 26 |
| 5.2 可改进的点 | 27 |

网上聊天室 FangTChat 说明文档

1 大程序简介

1.1 选题背景及意义

基于本学期对使用 Java 建立程序的学习, 使用 Java 进行完整的程序设计。在 eclipse 的编译环境下, 基于结构化程序设计思想, 运用数据结构等相关思想与有关概念方法, 应用 javafx 外部库, 针对程序设计题目与要求, 分析功能要求、划分功能模块, 设计功能模块函数, 以交互方式调用相应功能模块来实现信息处理。

本次程序设计的主题是网上聊天室, 根据所学的 jdbc 数据库以及程序设计知识, 实现对群聊\私聊\登录\注册等功能。系统针对交流中实际需要而设计, 能够方便交流。

1.2 程序设计目的

本课程设计选题为“网上聊天室”。系统主要功能是服务器转发信息, 实现

1、

①登录, 根据数据库读取信息, 判断密码是否正确

②注册用户加入到数据库.

③登录后通知其他客户端

④离开后通知其他客户端

⑤帮助指引, 便于使用者更快地掌握操作方法。

2、基于 javafx 的图形界面绘制, 包括文本框\密码框\按钮等组件

3、 所有用户需输入账号、密码登录进入系统; 进入系统后可查看其他用户是否在线, 和其他用户

私聊或群聊.

2 程序开发设计

2.1 程序设计原理

2.1.1 项目所使用到的模式

1.Adapter 模式

类适配器----我们使用 Myclient 来调用实现好的 FangTclient。修改 FangTclient 代码可能会出现无法意料的错误。这时我可以用 Adapter 模式去适配 FangTclient, 通过 Adapter 模式, 我可以继续多态地使用接口, 而避免因修改封装好代码而导致的错误, 不用考虑是如何实现 displayIt()的; 保证绝对不修改现有的类,只是在外边加一层代码.

2.对象适配器---委托模式

把各个按钮的处理交给各个 handler 类的实例,而不是 start 中创建匿名类来处理,有利于代码可读性.

3.Singleton 模式

保证任何情况下只有一个服务器实例.不会不小心用 new 创建实例,定义了获取唯一一个实例的 static 方法.如下所示

```
private static FtServer ftServer = new FtServer();
```

2.1.2 server 和 client 的登录协议

1. 启动服务器, 然后客户端启动,开始连接服务器, 服务器发送 `CONNECTSUCCESS`

2. 客户端接收 `CONNECTSUCCESS`, 等待客户端按按钮. 服务器等待 signal

```
int signal = fromClient.readInt();
```

3. 客户端 `toServer.writeInt(LOGIN);` 服务器接收 signal, `LOGIN`signal.

客户端发,服务器读

```
ftid = fromClient.readInt();  
String password = fromClient.readUTF();
```

然后 `toClient.writeInt(CONTINUESEND);`

然后客户端 `int i = receiveInfoFromServer();` 登录成功,就新建 `thread` 来和服务器 `while` 循环通讯.

2.1.3 server 和 client 的注册协议

1. 启动服务器, 然后客户端启动,开始连接服务器, 服务器发送 *CONNECTSUCCESS*

2. 客户端接收 *CONNECTSUCCESS*, 等待客户端按按钮. 服务器等待 signal

```
int signal = fromClient.readInt();
```

3. 按下按钮,客户端发注册信号 服务器接收 signal,
客户端发
服务器读

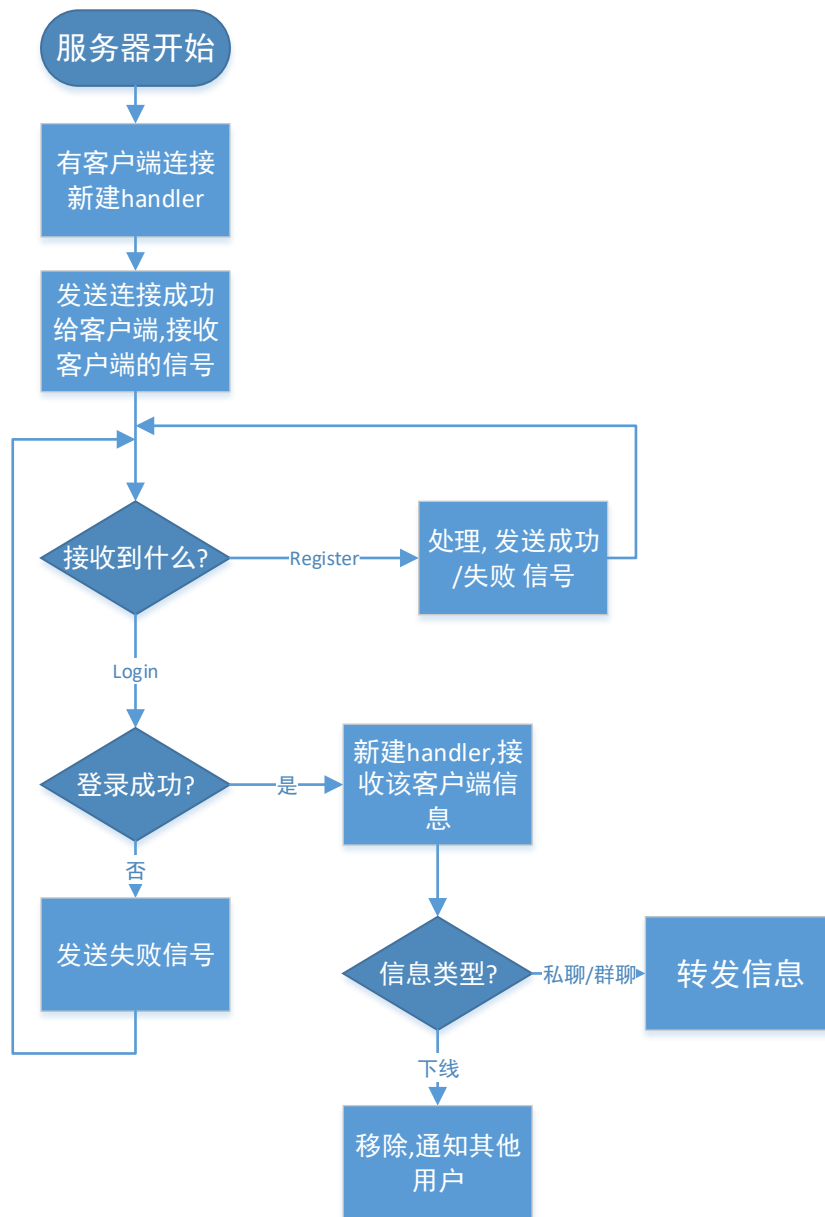
```
String username = fromClient.readUTF();
```

```
String password = fromClient.readUTF();
```

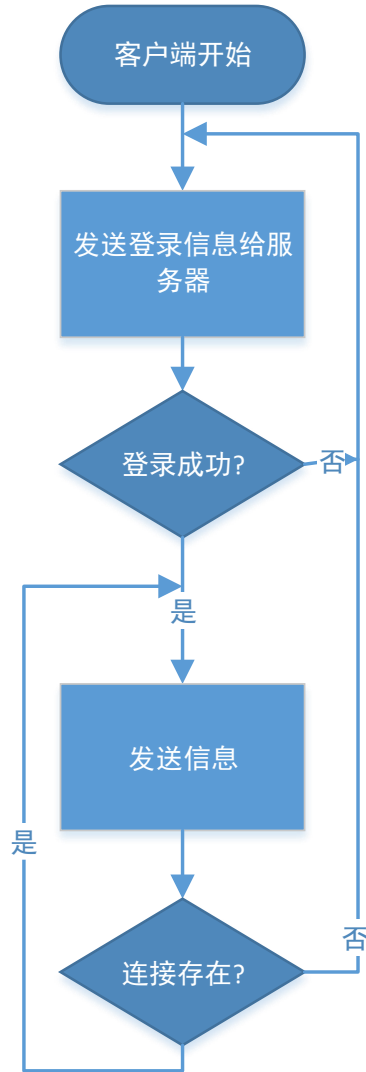
然后服务器不需要 `toClient.writeInt(CONTINUESEND);`

直接写一个 `String` 类型的返回信息给客户端即可,

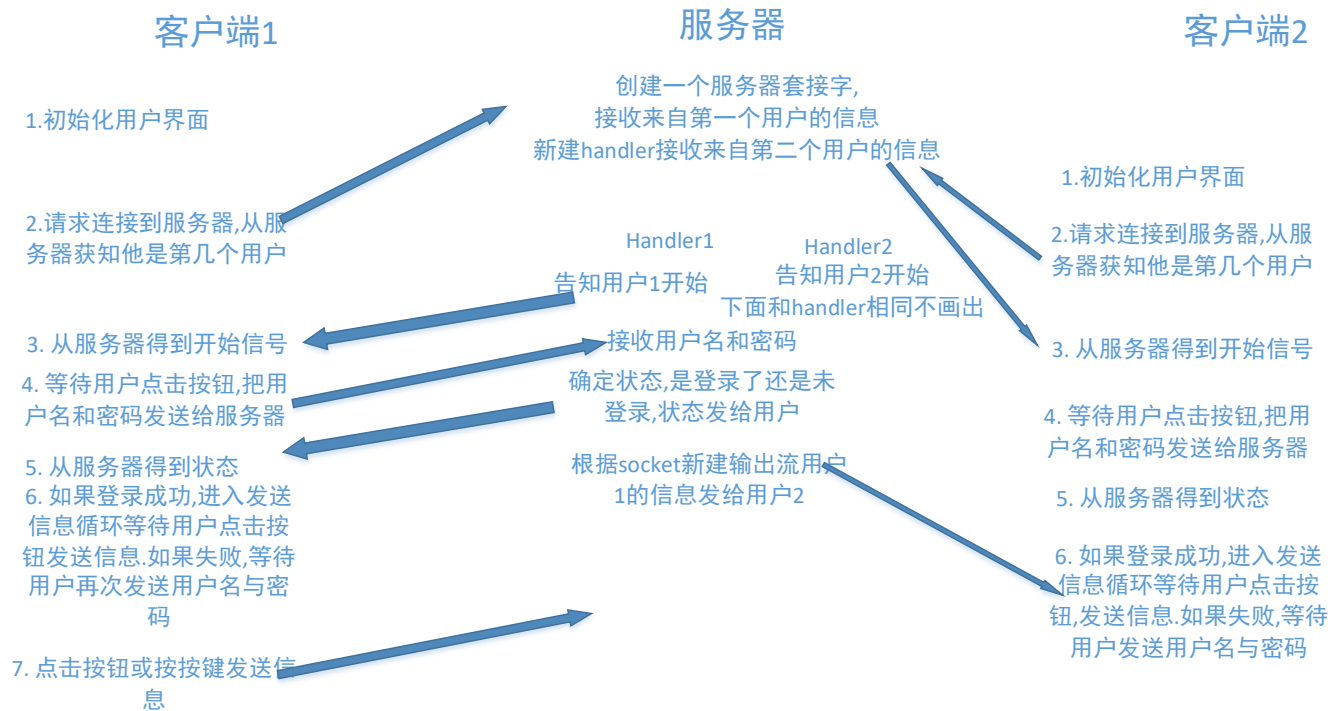
2.2 程序流程



服务器流程图



客户端流程图



2.3 文件列表

2.3.1 文件函数结构

FangTangConstants.java

```

v FangTangConstants
  > logrequest
    CONNECTFAIL
    CONNECTSUCCESS
    CONTINUESEND
    LOGIN
    port
    REGISTER
    SENDMESSAGE
    STOPSEND
  
```

| interface |
|---|
| <pre> +port: int = 9020 +CONNECTSUCCESS int = 11; +CONNECTFAIL int = 22; +STOPSEND int = 33; +CONTINUESEND = 44; +LOGIN = 55 +REGISTER = 66; </pre> |

FangTclient.java

| FtClient |
|--|
| +waiting: boolean +continueToSend: boolean +socket +fromServer: DataInputStream |
| +start(): void +adjustStyle(): void +resetByServerInfo(String) +receiveInfoFromServer() +waitForSendAction() +connectToserver() +BtnConnHandler() +BtnRegstHandler() +BtnSendHandler() +PressSendHandler() +close() +exit() |

FtServer.java

| FtServer |
|--|
| +users: ConcurrentHashMap +executorService: ExecutorService |
| +run(): void +Service(): void +sendToMembers(): void +sendToOne(): void +Handler() |

FtUser.java

```

    • FtUser()
    • FtUser(int, String, String)
    • add(String, String) : void
    • delete(int) : void
    • getAllName(int) : int
    • getFtid() : int
    • getName(int) : String
    • getNewestId() : int
    • getPassword(int) : String
    • getUsername() : String
    • Init() : void
    • setId(int, String) : void
    • setName(int, String) : void
    • setPassword(int, String) : void
    • toString() : String

```

MainL.java 启动客户端

Resource 文件夹,保存一些图标.

2.3.2 多文件构成机制

利用 interface , 把所需成员组合起来, 用来装封用到的常量,客户端和服务端都 implements constants 接口, 类似于 C 语言的头文件. 可以避免重复定义变量,实现类的多继承, 以解决 Java 只能单继承, 不支持多继承的问题。如下面两张图所示

```

// 定义了程序中所有类共享的常量.
public interface FangTangConstants {
    public static int port = 9020;
    public static int CONNECTSUCCESS = 11;
    public static int CONNECTFAIL = 22;
    public static int STOPSEND = 33;
    public static int CONTINUESEND = 44;
    public static int LOGIN = 55;
    public static int REGISTER = 66;
    public static int SENDMESSAGE = 67;
}

```

```

public class FangTclient extends Application implements FangTangConstants{
    Image imagesend = new Image("envelope.png");
    private Button btnExit=new Button("退出");
    private Button btnSend = new Button("发送",new ImageView(imagesend));
    private Button btConn = new Button("登录");
    private Button btRegst = new Button("注册");
    private TextField tfSend=new TextField();//输入信息区域
    private TextArea taDisplay = new TextArea();//显示区域
    private TextField username = new TextField();//填写用户名
    private PasswordField LOGINpassword = new PasswordField();//login in p
    private TextField fangTangId = new TextField();//填写id
    private PasswordField REGIpassword = new PasswordField();//填写密码
    private DataInputStream fromServer;
    private DataOutputStream toServer;
}

```

2.3.3 调用函数关系

登录

| 客户端任务 | 服务器任务 |
|------------------------------|-------------------------------|
| 发送账号密码BtnConnHandler | 接收账号密码，数据库中搜索 . responseLogin |
| 接收信息receiveInfoFromServer(); | 返回信息给客户端 |
| 更新按钮的状态resetByServerInfo | 登录成功则加入到hashmap |

注册

| 客户端任务 | 服务器任务 |
|------------------------------|--------------------------------|
| 发送账号密码BtnRegstHandler | 接收用户名密码，数据库中搜索 . responseRgstr |
| 接收信息receiveInfoFromServer(); | 返回信息给客户端 |
| 更新按钮的状态resetByServerInfo | 登录成功则加入到hashmap |

发送信息

| 客户端任务 | 服务器任务 |
|----------|---|
| 把信息写到显示栏 | 处理信息,比如通知其他客户,移除离线的客户端 responseSentence(message);或者sendToOne |

| | |
|---|-----------------------|
| 发送给服务器BtnSendHandler, PressSendHandler | 返回给客户端Client.writeUTF |
| 清空文本发送栏 | |

退出

| 客户端任务 | 服务器任务 |
|----------------------------|----------------------------|
| 发送字符串bye给ftserver, 让他移除掉用户 | 从map移除掉用户,该用户之后可以重新登录. |
| exit(),close(); | 通知其他用户该用户已经下线sendToMembers |

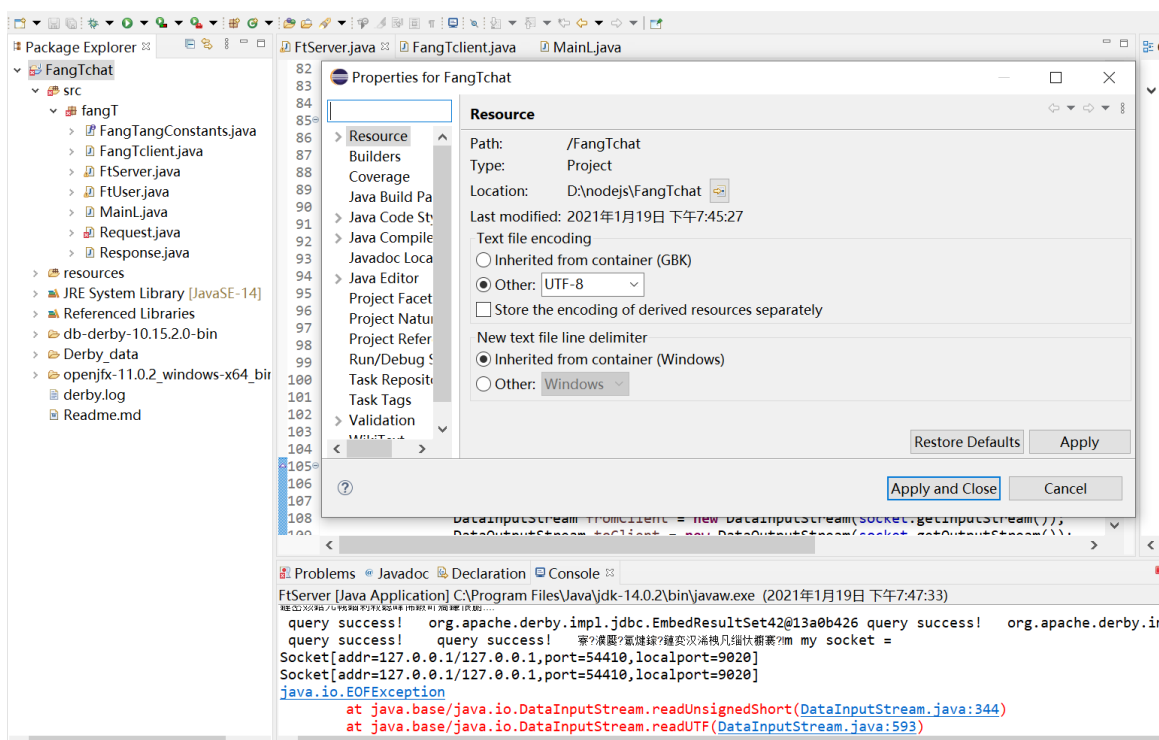
3 程序部署运行和使用说明

3.1 编译安装

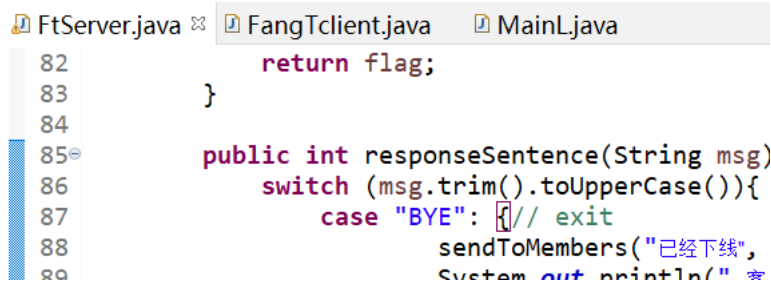
首先，解压压缩包

然后利用 eclipse 打开工程所在文件夹

注意可能中文乱码，请打开项目属性，如图所示用 UTF-8 打开



然后首先启动服务器程序

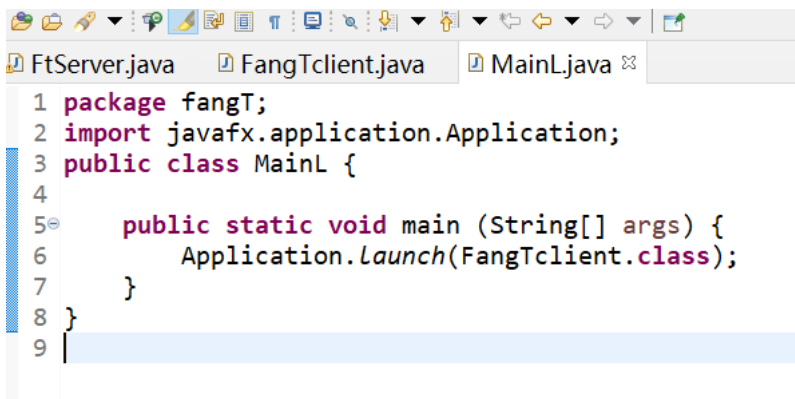


```

82         return flag;
83     }
84
85     public int responseSentence(String msg)
86     {
87         switch (msg.trim().toUpperCase()){
88             case "BYE": // exit
89                 sendToMembers("已经下线",
90                     System.out.println(" "

```

然后启动客户端程序,注意通过 MainL 启动,可以同时启动多个客户端,同时登录.



```

1 package fangT;
2 import javafx.application.Application;
3 public class MainL {
4
5     public static void main (String[] args) {
6         Application.launch(FangTclient.class);
7     }
8 }
9

```

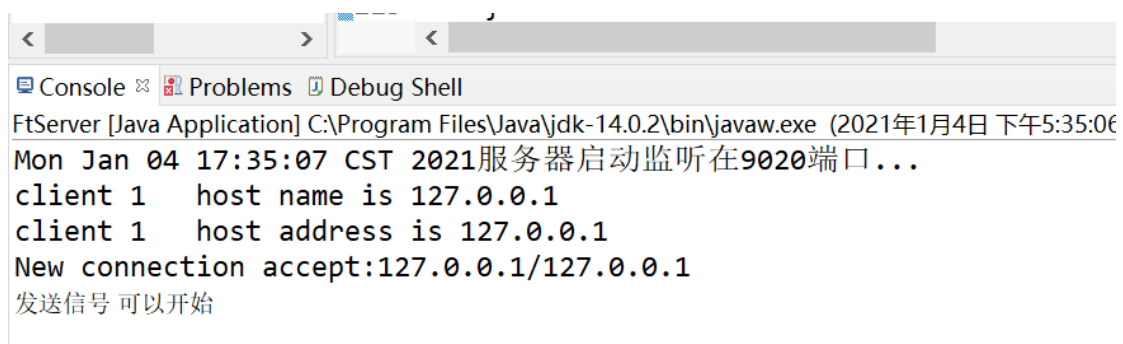
可以用下面几个已有账号测试

账号 202 密码 wes

账号 1 密码 123

账号 803 密码 123

3.2 实例运行测试



```

FtServer [Java Application] C:\Program Files\Java\jdk-14.0.2\bin\javaw.exe (2021年1月4日 下午5:35:06)
Mon Jan 04 17:35:07 CST 2021服务器启动监听在9020端口...
client 1 host name is 127.0.0.1
client 1 host address is 127.0.0.1
New connection accept:127.0.0.1/127.0.0.1
发送信号 可以开始

```

一个客户端连接时会显示它的 ip 地址



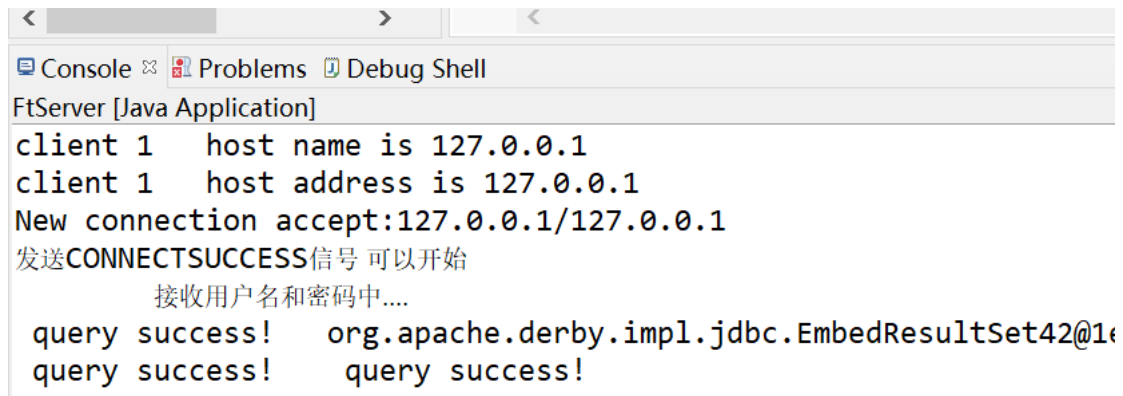
服务器接收用户名和密码中....

收到的用户名和密码为 = u5123

```
insert into ft_user(name,password) values( ? , ?)id = 2104name = u5passwrod = 123 user add suc
register success,没有用户名,注册了一个账户,用户名为 = u5账号为 2203
```

输入用户名 user6 和密码, 注册并返回一个自增不重复的账号.

输入用户名 user3 和密码, 从数据库中查询到已经存在该用户名,注册失败



输入账号 202,密码 wes, 服务器从数据库中查询到该 ftid 和 password 对应

信息显示区

```

连接服务器成功!!!
连接成功!!! 欢迎使用服务! 请输入用户名:
2021/01/10 13:08:22
202wes用户名和密码已经发送!
login in success! username is we 账号为 202
登录成功! 可以开始发送信息

```

客户端显示登录成功

```

连接成功!!! 欢迎使用服务!

```

```

请输入用户名:

```

```

2021/01/10 16:22:27

```

```

302123用户名和密码已经发送!

```

```

客户端返回login in success! username is user5 账号为 302

```

```

登录成功! 可以开始发送信息

```

```

本地客户端发送: en

```

```

收到服务器的消息: 收到的信息 = en来自

```

```

连接成功!!! 欢迎使用服务!

```

```

连接成功!!! 欢迎使用服务!

```

```

请输入用户名:

```

```

2021/01/10 16:20:31

```

```

1123用户名和密码已经发送!

```

```

客户端返回login in success! username is usertry

```

```

登录成功! 可以开始发送信息

```

```

本地客户端发送: java2021

```

```

收到服务器的消息: 收到的信息 = java2021来自

```

```

FtServer [Java Application]

```

```

Sun Jan 10 16:20:22 CST 2021服务器启动监听在9020端口...

```

```

client 1 host name is 127.0.0.1

```

```

client 1 host address is 127.0.0.1

```

```

New connection accept:127.0.0.1/127.0.0.1

```

```

发送CONNECTSUCCESS信号 可以开始

```

```

接收用户名和密码中....

```

```

query success! org.apache.derby.impl.jdbc.EmbedResul

```

```

query success! query success! 服务器收到信息 = java202:

```

```

client 2 host name is 127.0.0.1

```

```

client 2 host address is 127.0.0.1

```

```

New connection accept:127.0.0.1/127.0.0.1

```

```

发送CONNECTSUCCESS信号 可以开始

```

```

接收用户名和密码中....

```

```

query success! org.apache.derby.impl.jdbc.EmbedResul

```

```

query success! query success!

```

可以多用户登录,登陆后可以给服务器发送信息,

3.3 结果说明分析

3.3.1 初始界面

注册功能：注册,输入用户名密码。

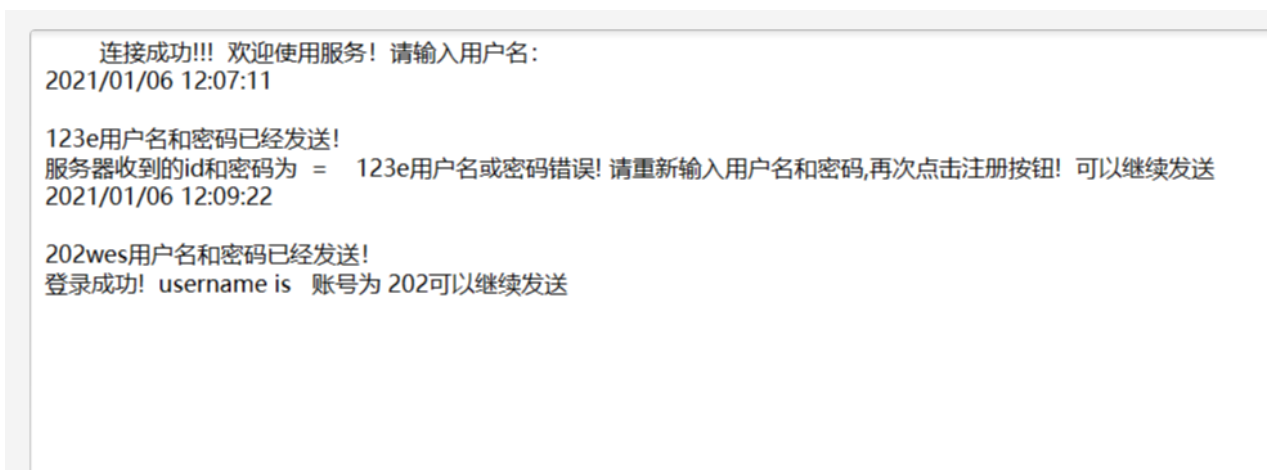


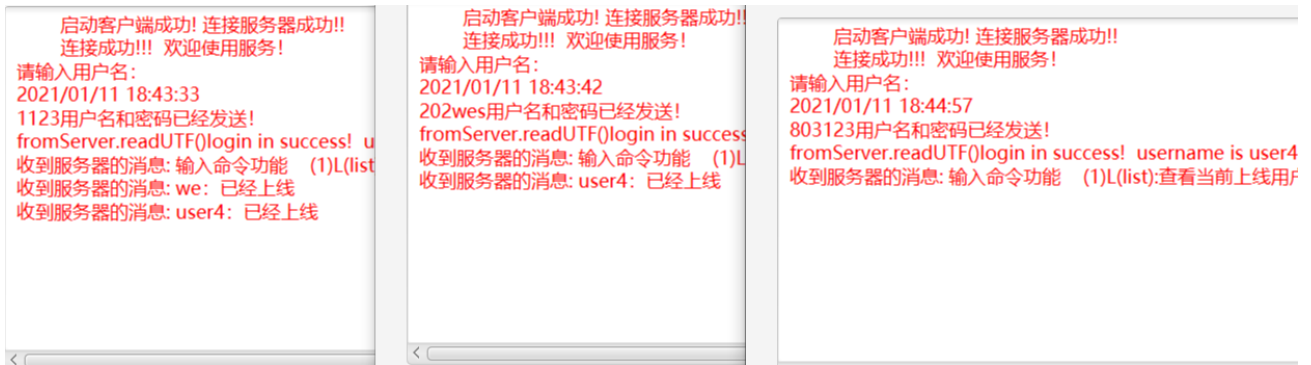
服务器接收用户名和密码中....

收到的用户名和密码为 = u5123

insert into ft_user(name,password) values(? , ?)id = 2104name = u5passwrod = 123 user add suc
register success,没有用户名,注册了一个账户,用户名为 = u5账号为 2203

登录：已经注册的账号即可直接登陆，输入用户名与密码即可。





实现了上线通知其他用户,图为 user4 上线,通知另外两个用户.

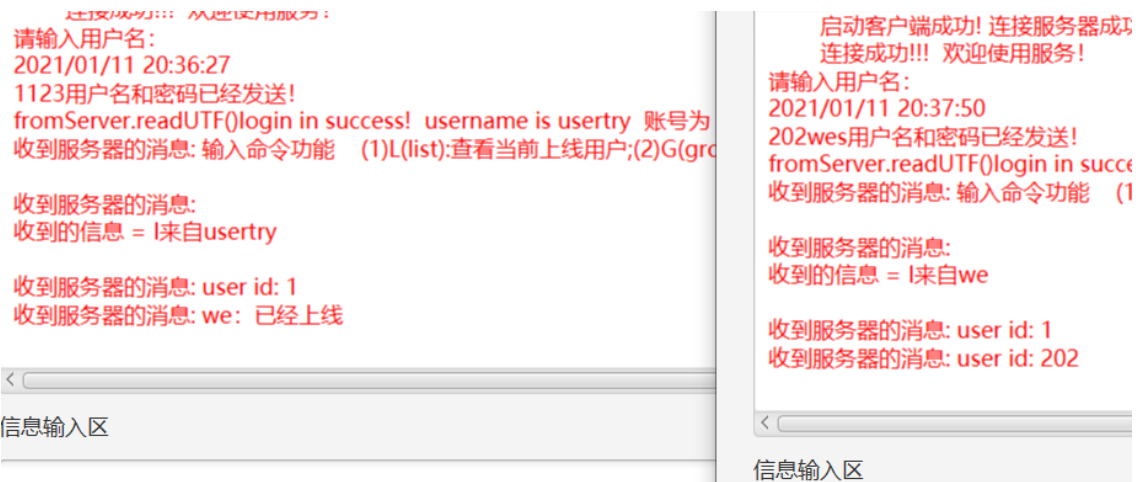
3.3.2 在线功能

(1)下线:客户端登录后输入字符串 bye 即可下线, 然后自动关闭客户端,可以重新登录 .

服务器会通知其他人,用 hashmap 来维护在线用户.



(2)查看所有在线用户的 id



3.3.3 发送信息功能

可以私信在线用户：



可以和在线用户群聊。

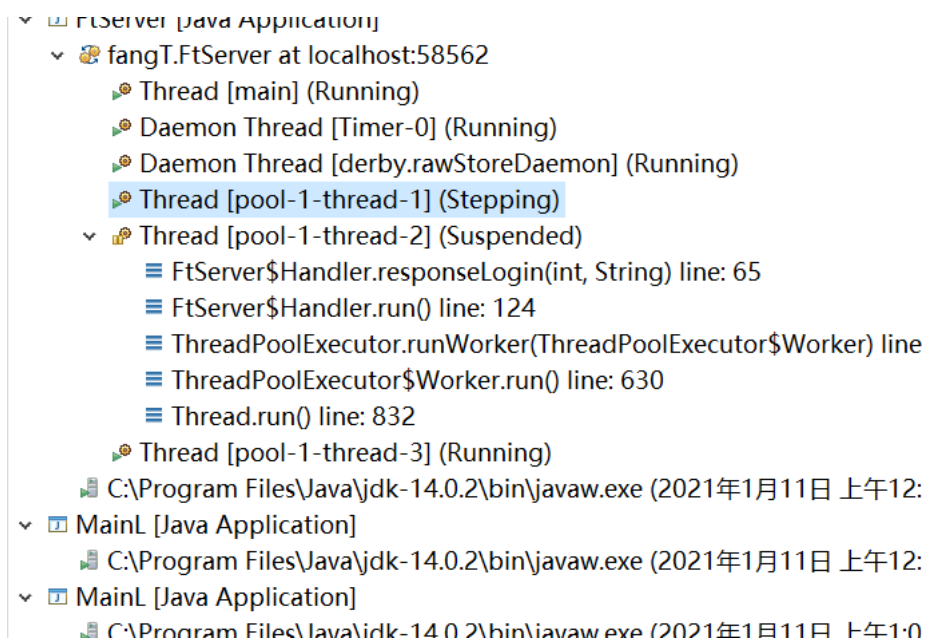
4 难点、要点、得意点

4.1 难点

4.1.1 多线程

发送信息可以通过发送按钮实现主动控制,但是对于输入流中有多少信息是不可控的,接收信息怎么不会卡住?再开一个客户端连接服务器,就会因为单线程阻塞程序卡住了。例如,如果服务器在一个客户端连接成功后,并没有一条信息发送给客户端,客户端的读取欢迎信息的语句无法读取到内容,就被阻塞住,由于是单线程,甚至整个程序都会被卡住。

解决方法:我们要让服务器和客户端互相约定通信规则,否则就可能有问题,Thread 类来实现客户端多线程接收,Runnable 类实现 run 方法,当登录成功的时候,我们新建一个进程,用来接收。我们用线程池,维护若干个线程,有新任务,分配一个空闲线程。我们并发小,可以采用动态调整线程池,实际中,线程用光了,可以用阻塞队列,阻塞队列满了,可能直接拒绝。否则会 out of memory OOM。把服务端和客户端通信的功能由一个线程来处理,就不会阻塞主进程的执行。



可以看到我们启动客户端有多个线程。

4.1.2 事件处理

在 start() 方法中注册注册事件处理程序。事件处理程序是事件处理程序接口的实现.例如

```
btRegst.setOnAction(new BtnRegstHandler());
```

class BtnRegstHandler implements EventHandler<ActionEvent>{}, 我们在类中重载 handle 方法

代码编写过程中,因为没有设置 key event 的 keycode == enter,导致按一个键就直接发送了.

4.1.3 服务器和客户端保持连接

怎么保证等待的时候, 客户端发送登录信号, 服务器可以正确接收?

用 http 协议, 可以用 java 后台给指定接口发送 json 数据.

用 TCP 也很简单,信号先行.任何时候他收到的都是信号. 两边约定好, 第一个发送的就是信号.

方法: 服务器 run(), writeInt(1); 告诉用户 1 可以开始了. 然后 while(true){

用户名 = datainput.readInt, 密码 = datainput.readString();

检查用户名和密码,

登录成功给他 writeInt(continue) 信号.// todo:然后转发给 2 他登录成功了.

信息 = fromuser1.readUTF.

再次判断状态.

如果 bye 那就退出

如果别的那就发给客户端.continue 信号, 然后发送 string.

然后循环.

}

但是我写了发现,其实没啥用, 全是字符串也完全可以.所以我的处理是登录后,while(), 互相发送字符串.

这里有几个点要注意,第1个问题是要正确区分长、短连接。所谓的长连接是一经建立就永久保持。短连接就是在以下场景下,准备数据—>建立连接—>发送数据—>关闭连接。要判断啥时候用长连接,啥时候用短连接。

第2个问题是对长连接的维护。所谓的维护包括两个方面,首先是检测对方的主动断连(既调用Socket的close方法),其次是检测对方的宕机、异常退出及网络不通。这是一个健壮的通信程序必须具备的。检测对方的主动断连很简单,主要一方主动断连,另一方如果正在进行读操作,则此时的返回值-1,一旦检测到对方断连,则应该主动关闭己方的连接(调用Socket的close方法)。

而检测对方,通常方法是用“心跳”,也就是双方周期性的发送数据给对方,同时也从对方接收“心跳”,如果连续几个周期都没有收到对方心跳,则可以判断对方或者宕机或者异常推出或者网络不通,此时也需要主动关闭己方连接,如果是客户端可在延迟一定时间后重新发起连接。

第3个问题是处理效率问题。不管是客户端还是服务器,如果是长连接一个程序至少需要两个线程,一个用于接收数据,一个用于发送心跳,写数据不需要专门的线程,当然另外还需要一类线程(俗称Worker线程)用于进行消息的处理,也就是说接收线程仅仅负责接收数据,然后再分发给Worker进行数据的处理。如果是短连接,则不需要发送心跳的线程,如果是服务器还需要一个专门的线程负责进行连接请求的监听。这些是一个通信程序的整体要求。

4.1.4 发送给其他客户端

思考了很久怎么让服务器主动发送给客户端,因为客户端之前并不知道服务器要发送?解决方法是用户连接时,立即向服务器发送自己的唯一ID,服务器端将ID和对应的socket用map存储.向客户端发送消息时,就利用之前保存的socket新建一个output.如下面的例子:

```
Socket tempSocket = (Socket) entry.getKey();
```

```
out=tempSocket.getOutputStream();
```

```
pw=new PrintWriter(new OutputStreamWriter(out,"utf-8"),true);
```

同一个客户端开两个 websocket 连接或者多个客户端连，其实效果是一样的，既然多个客户端可以连，一个客户端开多个连接也可以。

新开一个客户端 B ,出现问题有：

1. 本来是服务器发送给 A 信息，但是 A 收不到
2. A 发送信息，服务器可以接收到，但是返回给了 B，
3. A 继续发送信息，服务器不可以接收到.此时 B 发送，服务器也不可以接收到
4. 不能通知 A 已经上线，而是通知 B 已经上线

解决方法：

登录后新建一个 thread 接收。因为 inputstream 是全局变量,后一个登录了,全局变量就改了，所以别的客户端收不到。把 inputstream 放在 handler 里就可以了

4.2 编码规范

①基本要求

1.1 程序结构清晰,简单易懂

1.2 代码精简,直截了当,避免垃圾程序。

1.3 尽量使用标准库函数和公共函数,用 interface 和 implement, 高抽象，尽量可复用。比如我们不在一个 run 方法中写太多业务逻辑，而是尽可能分成函数来判断 message,然后返回 tag.

②可读性要求

2.1 保持注释与代码完全一致。

2.2 每个源程序文件,都有文件头说明。

2.3 每个函数,都有函数头说明,说明规格见规范。

2.4 主要变量(类或对象)定义或引用时,注释能反映其含义。

2.5 常量定义有相应说明。

2.7 处理过程的每个阶段都有相关注释说明。

2.8 利用缩进来显示程序的逻辑结构

示例如下:

```
//接收服务器发来的消息
private void receiveInfoFromServer() throws IOException{
//更新客户端的按钮等控件, 修改一些变量.
private void resetByServerInfo(String serverInfo){

// 把前端的工作分离出start();
public void adjustStyle(Stage primaryStage) {
// 按下注册按钮, 发送用户密码到服务器, 服务器去数据库找是否对应,如果有,那就显示已:
class BtnRegstHandler implements EventHandler<ActionEvent>{

class BtnConnHandler implements EventHandler<ActionEvent>{ //连接按钮触发的事件原:
// 按发送按钮
class BtnSendHandler implements EventHandler<ActionEvent>{
// 按下回车键
class PressSendHandler implements EventHandler<KeyEvent>{
private void waitForSendAction() throws InterruptedException, IOException {
    while (waiting) {
        Thread.sleep(100);
    }
    waiting = true;
}
private void exit() throws InterruptedException, IOException {
```

③变量命名

命名有一定的实际意义

具体例程:

```
public boolean continueToSend = true;
```

```
private TextField username = new TextField();//填写用户名
```

```
private PasswordField LOGINpassword = new PasswordField();//login in pwd
```

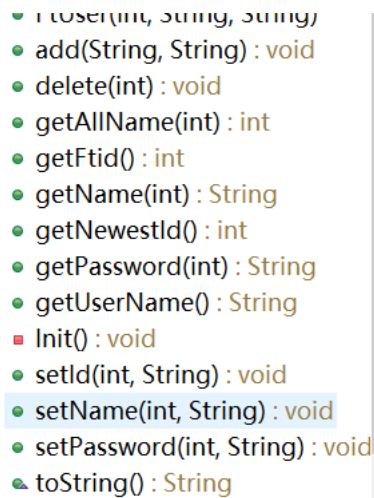
```
private TextField fangTangId = new TextField();//填写 id
```

④函数命名

函数原型说明包括引用外来函数及内部函数,外部引用必须在右侧注明函数来源:模块名及文件名,如

内部函数,只要注释其定义文件名;要求用大小写字母组合规范函数命名,必要时可用下划线间隔.

示例如下:



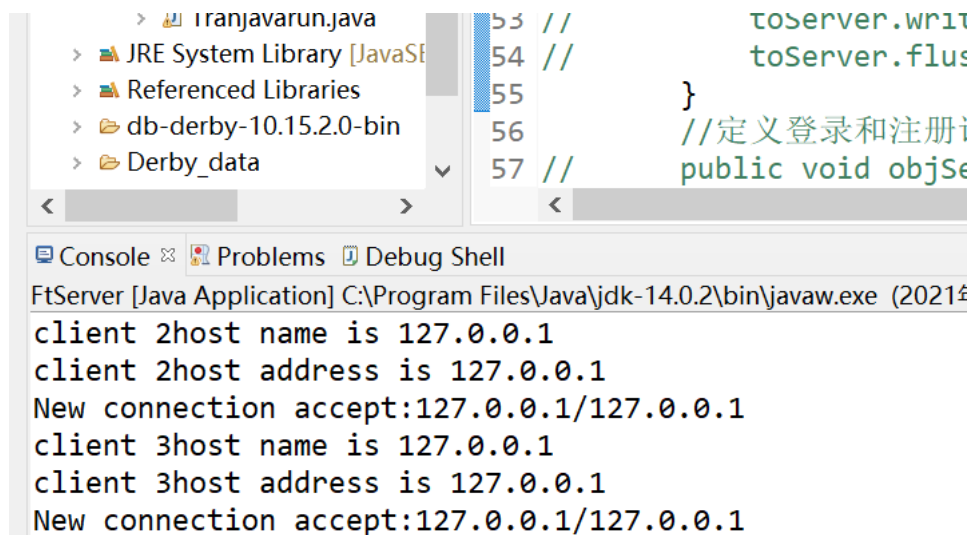
```
• setUser(int, String, String)
• add(String, String) : void
• delete(int) : void
• getAllName(int) : int
• getFtid() : int
• getName(int) : String
• getNewestId() : int
• getPassword(int) : String
• getUsername() : String
• Init() : void
• setId(int, String) : void
• setName(int, String) : void
• setPassword(int, String) : void
• toString() : String
```

4.3 要点

4.3.1 实现了多个客户端连接

将客户端接收信息的功能集中给线程处理，实现多线程同步进行。相当于多用户访问服务器资源，服务器应该与各个客户端建立连接，并进行通信对话，就像我们日常使用 QQ、微信、视频等客户端，就是多用户与服务器通信的例子。

服务端多线程的实现，目标是多用户（客户端）能够同时与服务器建立连接并通信，避免阻塞，进一步完善 TCP 的 Socket 网络通信，运用 Java 多线程技术，实现多用户与服务端 Socket 通信，而在多线程环境中，对共享资源的读写存在线程并发安全的问题，例如 HashMap、HashSet 等都不是线程安全的，可以通过 synchronized 关键字进行加锁，但还有更方便的方案：可以直接使用 Java 标准库的 java.util.concurrent 包提供的线程安全的集合。例如 HashMap 的线程安全是 ConcurrentHashMap，HashSet 的线程安全 Set 是 CopyOnWriteArraySet。



4.3.2 实现了和数据库的连接

```
connect 'jdbc: derby: dedb; user=db_user1; password=111111; create=true';
```

```
create table ft_user (ftid int NOT NULL GENERATED ALWAYS AS IDENTITY (START WITH 1,
INCREMENT BY 1), name varchar (20), password varchar (40));
```

```
insert into ft_user (name, password) values ('user1', '123');
```

```
select * from ft_user;
```

```
show ft_user;查看所有表
```

```
describe ft_user;
```

```
drop table try1;
```

```
select max(ftid) from ft_user;//命令行里面是可以的,返回刚插入的 id.但是 java 中会出错.
```

```
rs.getInt(1) 他是从 1 列开始的. 另外, resultset 要一行一行遍历.
```

注册输入用户名和密码,生成 ftid. 告诉用户 ftid, 用户通过 ftid 和密码来登录. Ftid 就像 qq 号. Name 就显示出来.

缺点: 有并发问题, 客户端需要注册, 关闭后再打开才能登录.

4.4 得意点

4.4.1 图标的位置

在阿里巴巴图标库找了图标, 设置了客户端的大 icon 和发送按钮的 icon.

4.4.2 异常的处理

退出时客户端做了处理, 一开始它有很多异常, 例如:

退出客户端时有 EOF 异常:

服务器也有 `java.net.SocketException: Socket closed`

然后都对这些异常进行了判断和处理, 让客户端可以正常退出不抛出异常.

4.4.3 前后端分离

在前后端分离方面的思考, 一个是利用服务器返回信息来判断并更新客户端 UI - `resetByServerInfo()`, 而不是在连接按钮中设置客户端的变化. 减少重复 `setbutton` 的代码, 增加程序的可读性.

另一个是 `adjustStyle` 方法 把前端的工作分离出 `start()`; 使用 `baseClient` 父类作为 UI 调整, 这样可以通过同样的 UI 写别的业务代码.

5 结论展望

5.1 心得体会

这次大作业让我受益颇丰. 最大的收获是加深了对 socket, 服务器和客户端多线程交互的理解, 让我知道了服务器和客户端交互有哪些技术难点, 比如如果都用 `writeint`, `writeUTF` 来定义通讯的每一步的话,

不仅冗长而且容易出错, 服务器多了一个 `writeInt` 可能客户端就收不到,进而明白了 http 数据包的巨大优势. 花费时间比较多的地方一个是曾经想把它改成 http 协议,但是由于对 http 认识不足,于是目前没有实现; 数据库和服务器的交互不是非常难, 做下来是比较顺利的一部分,因为之前已经实现了类操作数据库,也让我体会到了 java 类的好处.还学习了 javafx 的 GUI 和事件处理,之前一直不太明白事件处理和 while 循环到底是怎么一个顺序, 这次也在这个上面思考了很多时间,有了更深刻的理解.最后,感谢老师一个学期的教导!

5.2 可改进的点

1. 与客户端的连接方式, 可以改进为 HTTP 协议,加入解释器模式或者 Web Socket 协议,更加方便,不用登录的时候写 `writeInt`.
2. 一次登录 `bye` 后, 然后在客户端可以开启重新登录.
3. 服务器的 `run` 方法过长,有足足三页屏幕, 或许可以找到方法让他更易读.
4. 可以找一个更好的 GUI 框架. 现有的也可以创建一个列表,像 qq 一样,登录一个用户, 显示一个按钮,按下按钮即可通讯.现在的 UI 太不紧凑了.注册可以变成一个按钮然后弹出, 没必要 4 个输入框.
5. 下次可以先做完 ppt 展示, 再形成完整的报告.因为报告不如 ppt 那样可视化.