声　明

我声明，本软件课程设计报告及其研究工作和所取得的成果是本人（小组）独立完成的。研究过程中利用的所有资料均已在参考文献中列出，不存在整段或整句复制他人成果的情况，代码引用的部分已在报告和源文件中明确注明。其他人员或机构对本软件课程设计工作做出的贡献也已在致谢部分说明。

学生签名：许哲源

2018 年　1 月　25 日

LightChat 聊天系统设计总报告

1. 引言

在如今计算机高速发展的时代，用户对应用软件的要求也越来越高。控制台程序已经不能满足一般用户的需要，作为程序员应当学会为自己设计的的软件添加 GUI 界面，在完成软件功能的同时，尽可能的提升用户体验。而这次课程设计的目的，除了掌握 Socket 编程技术外，同时也应熟悉一门 GUI 编写语言。

2. 应用背景

本聊天系统适用于局域网中各 PC 客户端之间的通信与交流。

3. 需求分析

3.1 用户的需求

- 能够与其它在线用户进行沟通和交流；
- 在使用软件的过程中少出现程序阻塞等问题；
- 能自主决定何时开始通信和何时断开连接。

3.2 服务器的需求

- 具有较好的容错率，出现问题时可以及时处理；
- 能够对各客户端的信息进行记录，为各客户端间的通信提供桥梁；

4. 系统设计

4.1 模块划分

（1）客户端：

- 欢迎界面（Welcome Scene）：录入用户所用昵称
- 主界面（Primary Scene）：供用户聊天所用的界面

（2）服务器端

- 主界面（Primary Scene）：供服务器记录并显示日志的界面。

4.2 设计细节

- 使用多线程技术，实现了客户端间无阻塞 TCP Socket 通信，用服务器作为代理人处理来自不同客户端的请求与信息。

- 服务器可以在新用户加入时向客户端发出通知，并更新客户端用户列表。

  代码块：

```
for (int j = 1; j <= clientId ;j++) {
   DataOutputStream tempToClient = new
DataOutputStream(client[j].getOutputStream());
   tempToClient.writeUTF("/arg  refreshAliveUsers");
   tempToClient.flush();
   for (int i = 1; i <= clientId; i++) {
      tempToClient.writeUTF(clientInfos[i].getNickName());
      tempToClient.flush();
   }
   tempToClient.writeUTF("/end");
   tempToClient.flush();
   if (j!=clientId)
      tempToClient.writeUTF("from server : "+clientInfos[clientId].getNickName()+" is
online!");
}
```

- 使用转义字符串 “ /arg ” 实现系统命令行的传送，从而保证各项特殊功能的实现。

  代码块：

```
String info=fromServer.readUTF();
if (!info.startsWith("/arg"))
   Platform.runLater(()->{
      ta.appendText(info+"\n");
   });
else {
   String arg=info;
   if (arg.contains("getId")) {
      id = fromServer.readInt();
   } else if (arg.contains("clear")) {
     ta.clear();
   } else if ( arg.contains("refreshAliveUsers")) {
      users.clear();
      arg=fromServer.readUTF();
     ……//其它代码  }
```

- 对连接请求设置了超时限制，让客户端在对方无应答时可以及时返回。

  代码块：

```
client2.setSoTimeout(20000);
while (bool) {
 … … //其它语句
} catch (IOException ex) {
   try {
      toClient1.writeUTF("from server : Sorry, connection request Time Out, please try
later.");
      toClient2.writeUTF("from server : Time Out !");
      toClient2.writeUTF("/arg   closeRequestStage");
      clientInfos[client1Id].setInChat(false);
      clientInfos[client2Id].setInChat(false);
      client2.setSoTimeout(0);
   } catch (IOException ex2) {
      ex.printStackTrace();
   }
      continueToChat = false;
      return;
}
```

- 当被连接对象正在会话中时，系统拦截请求，并向被连接对象发出通知。

  代码块：

```
if (clientInfos[requestToId].isInChat()) {
   toClient.writeUTF("Sorry, "+clientInfos[requestToId].getNickName()+" is busy now,
please try later.");
   DataOutputStream tempOut = new
DataOutputStream(client[requestToId].getOutputStream());
   tempOut.writeUTF("from server : "+clientInfos[clientId].getNickName()+" try to
connect you just now, you can handle this later.");
}
```

- 对系统安全性进行了考虑，禁止用户在文本框中直接对服务器发出命令。

  代码块：

```
TextField textWantToSend = new TextField();
textWantToSend.setOnAction(event -> {
   if (textWantToSend.getText().contains("/arg")) {
      ta.appendText("Warning: /arg is used by system, you can not include it in
text.\n");
   } else {
      ta.appendText(nickName + ": " + textWantToSend.getText() + '\n');
      try {
         toServer.writeUTF(textWantToSend.getText());
         toServer.flush();
      } catch (Exception ex) {
         ta.appendText(ex.toString() + '\n');
      }
```

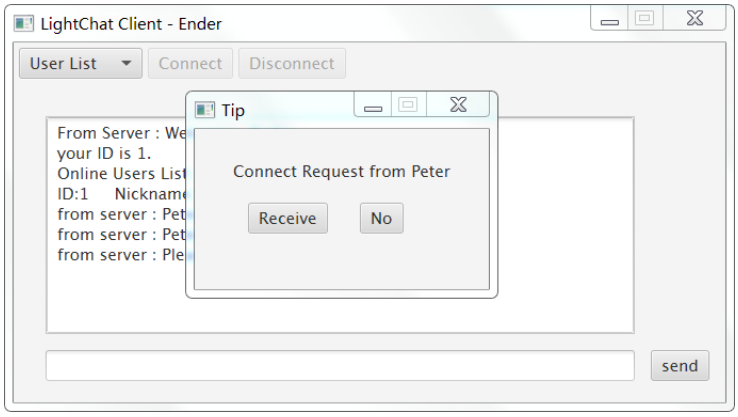
```
    }
    textWantToSend.clear();
});
```

- 当用户接收到其它客户端的连接请求时自动弹出连接请求窗口。

  代码块：

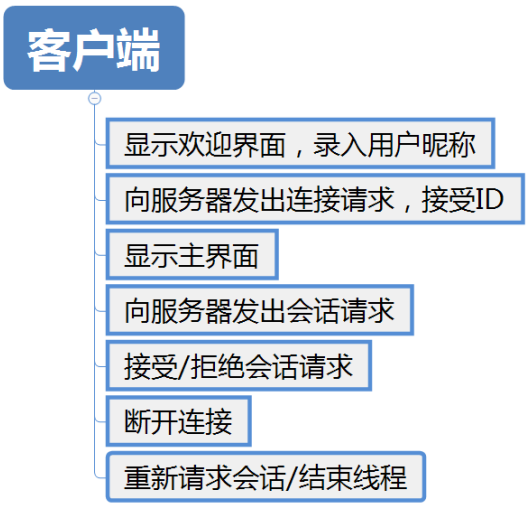
```
Label lbTip = new Label("Request");

HBox hBoxRequest = new HBox(btReceive,btNo);
hBoxRequest.setSpacing(30);
hBoxRequest.setPadding(new Insets(20));
BorderPane paneRequest = new BorderPane();
paneRequest.setTop(new StackPane(lbTip));
paneRequest.setCenter(hBoxRequest);
Scene sceneRequest = new Scene(paneRequest);
paneRequest.setPadding(new Insets(30,30,20,30));

stageRequest.setTitle("Request");
stageRequest.setScene(sceneRequest);
stageRequest.setFullScreen(false);
stageRequest.setResizable(false);
stageRequest.setAlwaysOnTop(true);

if (arg.contains("connectRequest")){
    ta.appendText(fromServer.readUTF());
    String nameRequest = fromServer.readUTF();
    Platform.runLater(()-> {
        stageRequest.setTitle("Tip");
        lbTip.setText("Connect Request from "+nameRequest);
        stageRequest.show();
    });
```

  运行效果图：



5.项目开发

5.1 系统流程图



5.2 关键模块的实现技术

客户端：采用简约的 JavaFX 设计技术和 Socket 编程技术；

服务器端：采用多线程技术，以编程实现接口；

（程序代码在附件中呈现）

5.3 特殊问题及解决办法

（1）没有语言基础和 Socket 编程知识

- 问题描述：

对于个人而言，实现该系统最大的问题在于缺乏相关的知识基础和实践经验，没有接触过任何 GUI 设计语言和 Socket 编程技术，在开始时便无法对这个系统有一个完整流程的把握。

- 解决方案：

从网络上搜集相关项目资料，了解实现该系统大体流程，之后确定实现该系统所需要的知识和技巧，通过对每项需要用到的技术分别进行学习和实践，最终用 Java 语言实现了这个系统，学到了很多新的知识与编程技巧。

（2）程序在运行时的阻塞问题

- 问题描述：

这个问题在开始时困扰了我好久，不知道在哪里发生了阻塞，后来通过调试发现：
按照构想，服务器（Server）为监听每个客户端的而新建一个线程，而当一个客户端发起连接请求时，只有发起请求的一方线程可以及时返回，而被请求的那一方的线程会因 readUTF（）方法而阻塞无法及时返回，这就导致接下来的会话无法继续。

- 解决方案：

让服务器在接受到会话请求时，先让请求方的对应的线程 sleep，再让被请求的客户端返回一条命令行信息（”/arg preForConnection”）,从而使其在服务器线程中的对应线程能够返回，待两个线程都为会话请求准备好时再建立新线程处理会话。

- 代码块：

服务器端：

```
if (info.startsWith("/arg")) {
   if (info.contains("connectRequest")) {
              …… //其它代码
      } else {
         clientInfos[clientId].setInChat(true);
         DataOutputStream tempOut = new
DataOutputStream(client[requestToId].getOutputStream());
         tempOut.writeUTF("/arg preForConnection");
         Thread.sleep(500);
         new Thread(new HandleSession(client[clientId], client[requestToId], clientId,
requestToId)).start();
      }
   } else if(info.contains("preForConnection")) {
```

```
        clientInfos[clientId].setInChat(true);
        toClient.writeUTF("/arg preOkay");
```

客户端

```
if (arg.contains("preForConnection")) {
    toServer.writeUTF("/arg preForConnection");
}
```

5.4 工作周志

- 第一周：

明确选题，考察实际需求，确定软件要实现的具体目标。

通过书本了解 JavaFX 中 Stage、Scene、pane 和 node 的概念，理解之间的联系。

学习 Socket 编程在 java 中的实现方式，通过做小程序来掌握各种特性。

理解了多线程的概念和在 java 实现方式，通过 lambda 表达式来简化代码。

- 第二周：

设计客户端 GUI 界面，完成客户端的基本功能。

调试系统在实现过程中出现的各种 Bug.

为解决阻塞问题而设计出了服务器与客户端间的异步应答方式，

保证了在系统运行的过程中不会出现线程间的冲突。

6.个人小结

这次课程对于我不仅是一次实践，更是一种挑战。从这次的经验中，我学到了如何从零开始完成一个没有知识背景的项目，我想这就是参加这次课程最大的收获。当然，也学会了很多实践所得到的专业知识，学会了如何用 Socket 来编写通信程序，以及如何用 JavaFX 来编写 GUI 程序。相信这些知识会在以后具体接触这些课程时，会成为我深入探究的动力。

7.参考资料

1. Y.Daniel Liang.《Java 语言程序设计（基础篇）》.第 10 版.机械工业出版社,2015-7

2. Y.Daniel Liang.《Java 语言程序设计（进阶篇）》.第 10 版.机械工业出版社,2016-9