

Java 期末项目文档

实现简单的多人聊天器

2013/2/22

同济大学软件学院

赵青 1152683

Java 期末项目文档

赵青 1152683

一. Run Instruction

Note: According to the assignment paper, only one `chat_server_tcp_port` is needed, however, when implementing the chat server, another **`chat_server_udp_port`** (to which the client send messages though udp socket) is necessary as well. To avoid hard coding, **there should be two port parameters as shown below in yellow shadow.**

(一) Run Sample

Server:

Parameter: `<chat_server_tcp_port> <chat_server_udp_port>`

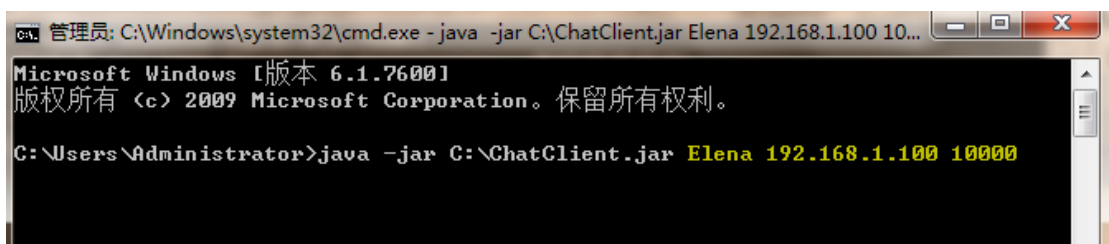


```
管理员: C:\Windows\system32\cmd.exe - java -jar C:\ChatServer.jar 10000 20000
Microsoft Windows [版本 6.1.7600]
版权所有 (c) 2009 Microsoft Corporation。保留所有权利。

C:\Users\Administrator>java -jar C:\ChatServer.jar 10000 20000
服务器已启动.
```

Client:

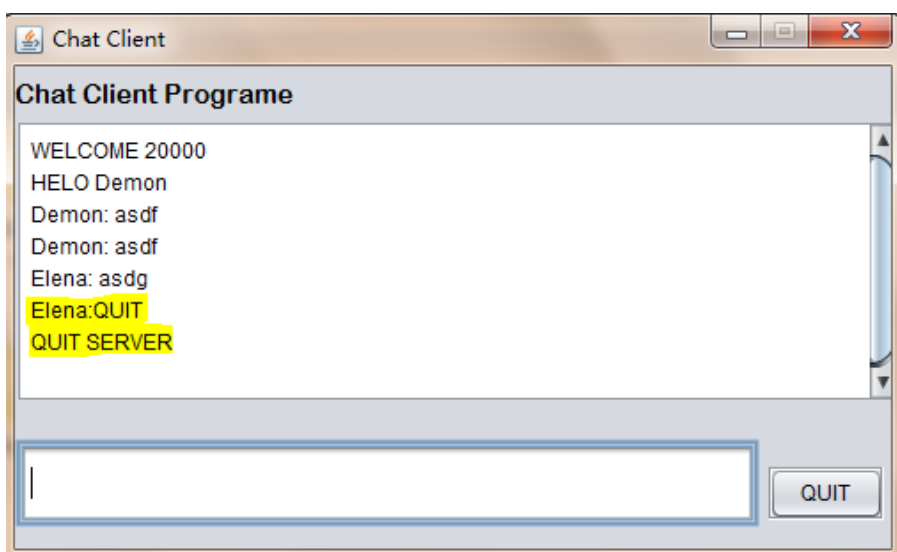
Parameter: `<screen_name> <chat_server_hostname> <chat_server_tcp_port>`



```
管理员: C:\Windows\system32\cmd.exe - java -jar C:\ChatClient.jar Elena 192.168.1.100 10000
Microsoft Windows [版本 6.1.7600]
版权所有 (c) 2009 Microsoft Corporation。保留所有权利。

C:\Users\Administrator>java -jar C:\ChatClient.jar Elena 192.168.1.100 10000
```

(二) User Interface



二. Design Ideas

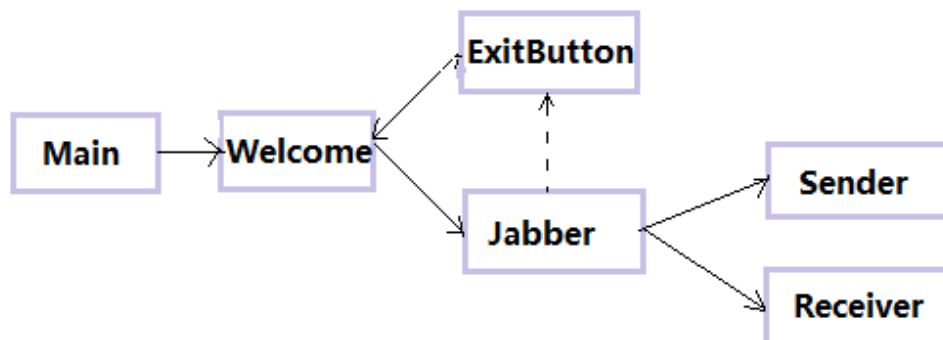
To accomplish the server's job, I divide it's duty to four parts: connection setup, message receiving(UDP) , message processing(relaying), and sending(TCP), thus four functional class (introduced below) is needed, plus two auxiliary classes for GUI and main method. To store the client sockets, at first I chose to use **ArrayList** at the beginning, but later I found it's hard for removing, **HashMap** is hands down more appropriate :p...

As for client, it's job are connection setup and messages sending &receiving based on the GUI interface. According to the requirement, the client is not multi-threaded. Thanks to the event handling mechanism in Java the client could work asynchronously, but to show the messages, I use the elements(should be private but I had to make it public anyway) in GUI as the parameters of Connection, dunno if it's acceptable...

三. Class Functionality Description

(Note: Only for **simple demonstration** of the basic relationship between classes, **solid arrow** for aggregation while **dotted arrow** for comparatively indirect relationships between user interface and implementation methods)

(一) Chat Server



Class Description

Main: Creates an instance of Welcome to start the server;

Welcome: Launches the server, creates the jabber thread, waits for connections from clients on the TCP Welcome Socket and complete the following job;

ExitButton: Creates a JButton to exit the server and send the "QUIT SERVER" message;

Jabber: Continuously reads messages from the UDP socket and sends them to all the clients over their respective TCP sockets.

Sender: Sends the message(bytes) to all clients in the **ArrayList (required data structure)** via tcp socket;

Receiver: Reads messages from the UDP socket

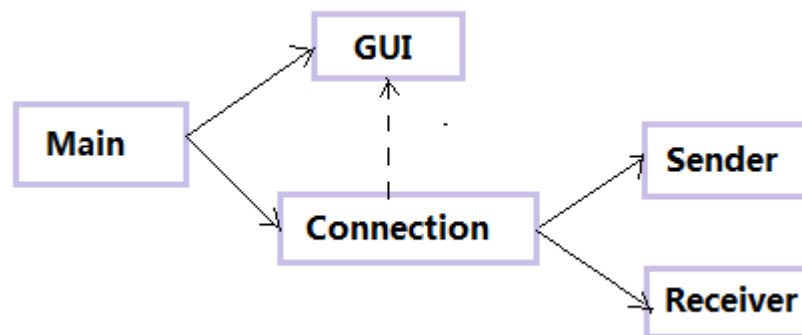
(二) Chat Client

Class Description

Main: Creates GUI and an instance of Connection to start the client;

GUI: Provide the user interface;

Connection: Accomplish all the required functionality of a client using sender and receiver



along with certain methods inside;

Sender: Sends message to the server over UDP;

Receiver: Reads TCP socket for messages from the server.

四. Merits and Demerits analysis

Merits:

1. Successfully implements a chat server to which multiple clients connect and client able to communicate with the server, and all the features are implemented as described in the protocol;
2. Break up the code into separate classes to demonstrate basic OOP skills with Java
3. Error handlings are implemented where I think it appropriate.
4. Sufficient code documentation for readability.

Demerits:

1. In the client part, to show the messages, I use the elements like JTextArea which should be private but ended up public in GUI as the parameters of Connection. This breaks the rule of encapsulation of OOP;
2. Duplication of name isn't handled in my program cause of limited time, will try to revise it later, though too late...

五. 项目心得

(想来想去决定这部分还是用中文写, 还是母语才能更好的表达自己的感受。。。)

由于各种过年综合症, 项目开始的比较晚, 现在回想整个过程, 并不如想象中的顺利。

首先整个功能的理解和建立在此上的类的功能划分就花去了不少的时间, 这次遗憾的是并没能很好地运用类的继承, 仅仅停留在简单的新建类的实例上。

而后开始根据老师最后一节课的笔记自学 java 网络编程, 并开始一点点的写自己的项目。由于网上相关的教材很多, 开始进展的比较顺利, 但到了调试阶段, 各种连接问题接连不断, 这也是我整个项目过程中最痛苦的时候, 期间毫不夸张的说我几乎把所有网络编程的常见异常都处理了一遍, "EOFException", "connection refused connect. socket is closed", "connection reset", "connect reset by peer. broken pipe", "BindException:address already in use"... ... 但事后却觉得收获很大, 甚至在有些地方又把上学期学的计算机网络深入复习了一下, 同时又一次感慨代码看一遍和自己写一遍真的天差地别啊!

在服务器和客户端终于能通信后, 剩余的工作相对较为顺利, 可是由于之前的问题占用的时间过长, **有些细节方面难免处理的比较粗糙**(部分问题在上面的 Design idea 和 Demerits A nalysis 中有所说明), **望见谅。**

做这个项目虽然中间的过程比较痛苦, 但在完成后再次回首, 发现真的学到了很多, OOP 的基本思想, java 编程能力和计算机网络方面都有让自己欣喜的提高, 同时更加意识到自己知识和能力方面的诸多不足和漏洞。Java 这门课虽然到这里告一段落, 但我的学习从未停止, 希望在接下来的日子里通过自身的努力不断进步!