****

**期末考核大作业报告**

作业成绩：

课 程：程序设计（4）（Java平台）

姓 名： 何庚坤

学 号： 2018214139

班 级： 13901801

**目录**

[1 大作业1 1](#_Toc43034458)

[1.1 作业题目 1](#_Toc43034459)

[1.2 程序分析与设计 1](#_Toc43034460)

[1.3 程序关键代码及解释 2](#_Toc43034461)

[1.4 系统运行截图 5](#_Toc43034462)

[1.5 总结与心得 7](#_Toc43034463)

# 1 大作业1

## 1.1 作业题目

基于多线程模式的流式套接字通信应用程序

## 1.2 程序分析与设计

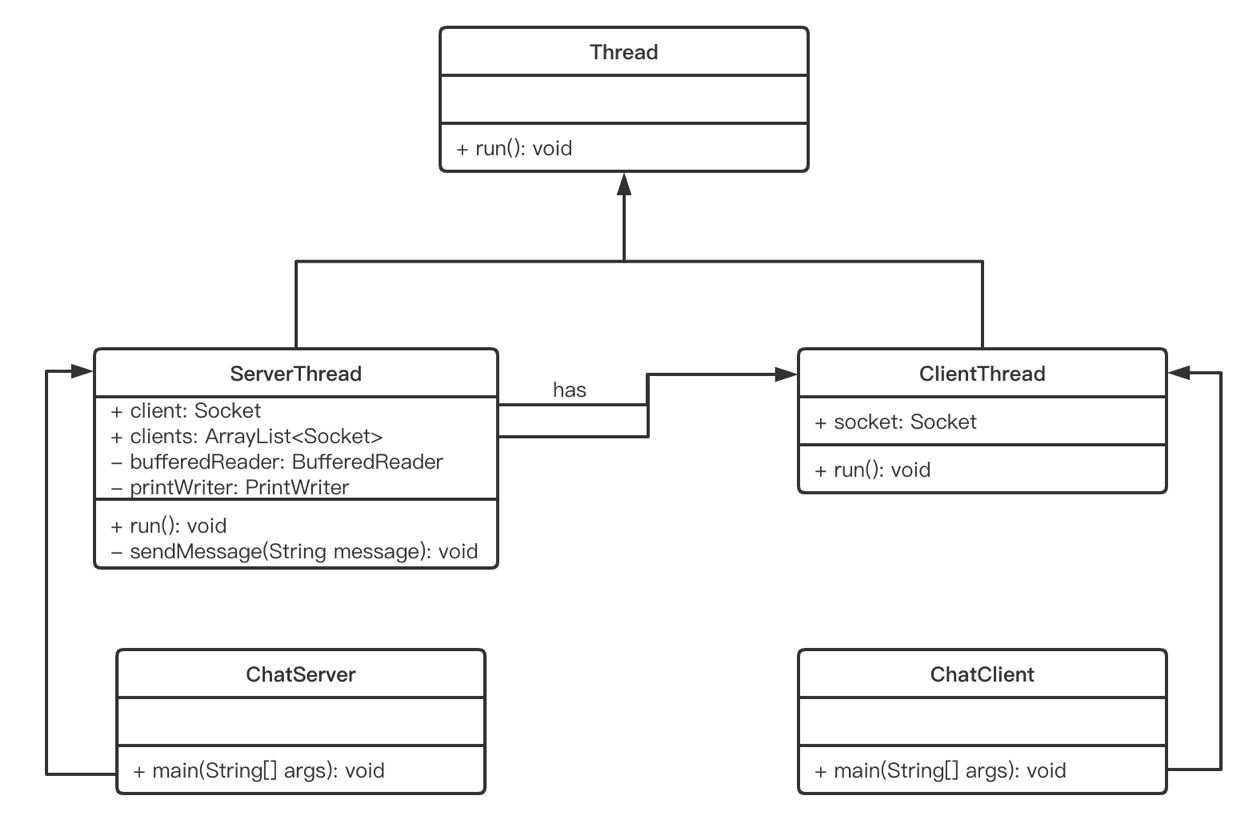
**1、程序设计思想**

本系统使用多线程模型实现，服务端和客户端之间通过Socket进行连接通信，客户端通过命令行输入要发送的信息，服务端收到后将信息发送给所有客户端，完成广播

**2、程序功能介绍**

* **进入聊天室：**运行客户端后即可建立服务端和客户端之间的连接
* **多人聊天：**通过在客户端发送信息，回车发送后即可通过服务端广播发送给所有客户端，使所有客户端都收到信息

**4、类设计**

****

**ChatServer：**在main中创建ServerSocket服务，while(true)循环监听客户端连接，每连接一个客户端Socket就创建一个ServerThread进行处理

**ServerThread：**在初始化时，创建对应的客户端的Socket和所有客户端Socket的列表，当客户端发来信息时，对应的ServerThread处理信息，并通知给所有客户端，进行广播

**ChatClient：**在main中创建客户端的Socket，同时开启ClientThread处理Socket，while(true)获取BufferedReader获取的键盘输入

**ClientThread:** 通过InputStreamReader获得输入流，得到BufferedReader后通过PrintWritter发给Socket的OutputStream发给服务端

## 1.3 程序关键代码及解释

class ServerThread extends Thread {

public Socket client;

public ArrayList<Socket> clients;

private BufferedReader bufferedReader;

private PrintWriter printWriter;

// **初始化对应客户端Socket和所有的客户端ArrayList<Socket>**

ServerThread(Socket client, ArrayList<Socket> clients) {

this.client = client;

this.clients = clients;

}

@Override

public void run() {

try {

// **读客户端的输入**

bufferedReader = new BufferedReader(new InputStreamReader(client.getInputStream()));

// **获取当前客户端的index，以此区别不同的客户端**

int user = clients.indexOf(client);

String massage = "Welcome user " + user + " !!!";

sendMassage(massage); // **客户端刚连接时发送欢迎信息**

String chatMsg;

while (!"exit".equals(chatMsg = bufferedReader.readLine())) {

chatMsg = "User " + user + " : " + chatMsg;

sendMassage(chatMsg); // **不断读取客户端输入进行发送信息**

}

} catch (IOException e) {

e.printStackTrace();

}

}

private void sendMassage(String massage) {

try {

System.out.println(massage);

// **遍历进行广播**

for (int i = 0; i < clients.size(); i++) {

// **获取客户端的prinntWriter打印到客户端命令行界面**

printWriter = new PrintWriter(clients.get(i).getOutputStream(), true);

printWriter.println(massage);

printWriter.flush();

}

} catch (IOException e) {

e.printStackTrace();

}

}

}

public class ChatServer {

public static void main(String[] args) {

try {

// **创建服务端Socket**

ServerSocket serverSocket = new ServerSocket(7000);

ArrayList<Socket> clients = new ArrayList<Socket>();

System.out.println("聊天服务器启动，端口：7000。。。");

while (true) {

// **每个新的客户端都创建一个对应的ServerThread进行处理，并添加到clients中**

Socket client = serverSocket.accept();

clients.add(client);

new ServerThread(client, clients).start();

}

} catch (IOException e) {

e.printStackTrace();

}

}

}

class ClientThread extends Thread {

Socket socket;

// **初始化Socket**

ClientThread(Socket s) {

socket = s;

}

@Override

public void run() {

try {

// **BufferedReader读客户端输入，PrintWriter进行输出**

BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));

PrintWriter writer = new PrintWriter(socket.getOutputStream(), true);

String massage;

// **不断读取客户端输入，并打印**

while (true) {

massage = reader.readLine();

writer.println(massage);

}

} catch (IOException e) {

e.printStackTrace();

}

}

}

public class ChatClient {

public static void main(String[] args) throws IOException {

// **创建socket连接**

Socket socket = new Socket("127.0.0.1", 7000);

// **开启ClientThread处理**

new ClientThread(socket).start();

BufferedReader reader = new BufferedReader(new InputStreamReader(socket.getInputStream()));

String massage;

while ((massage = reader.readLine()) != null) {

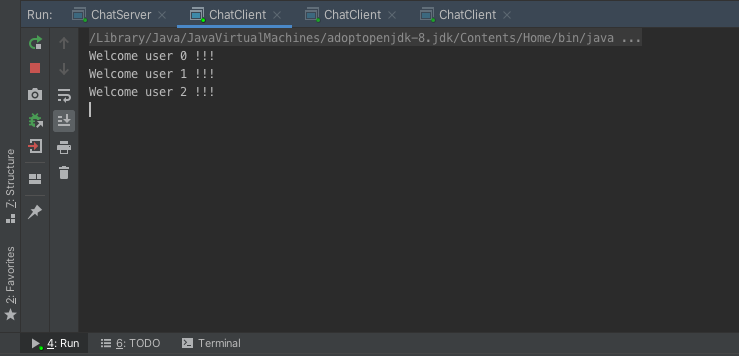
System.out.println(massage);

}

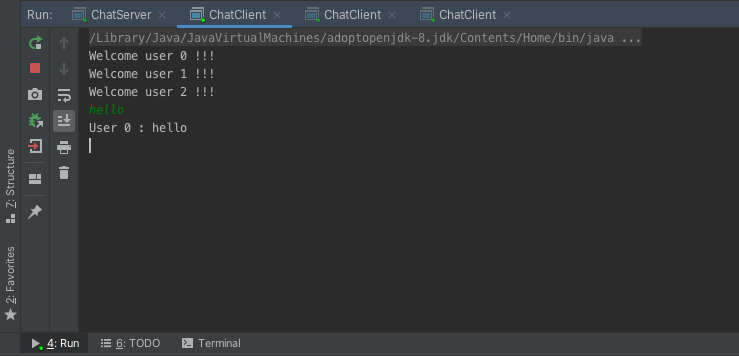
}

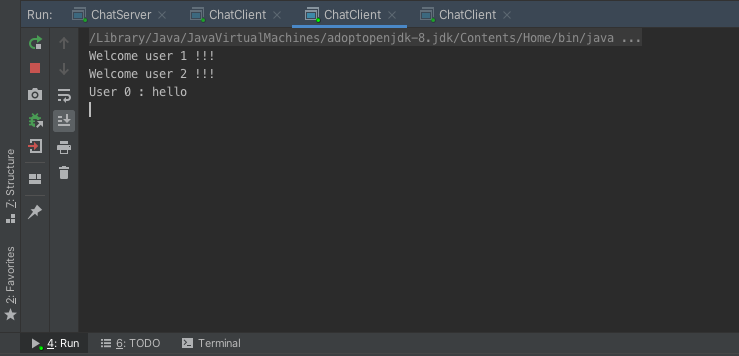
}

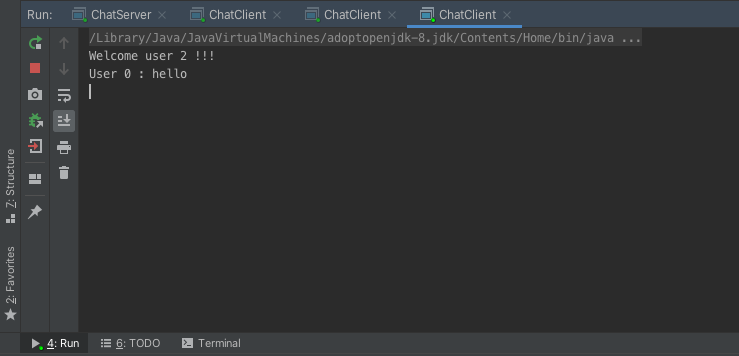
## 1.4 系统运行截图

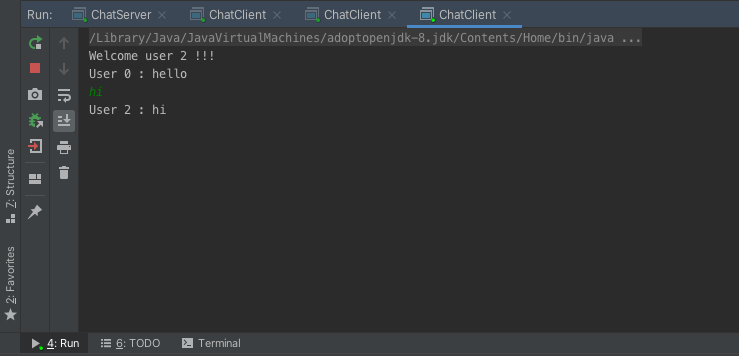
****

创建服务端和三个客户端

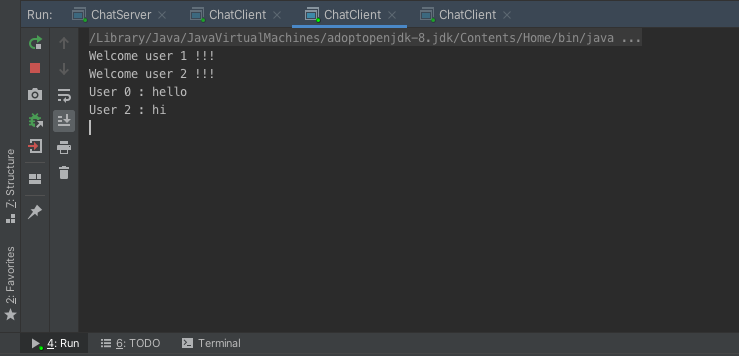


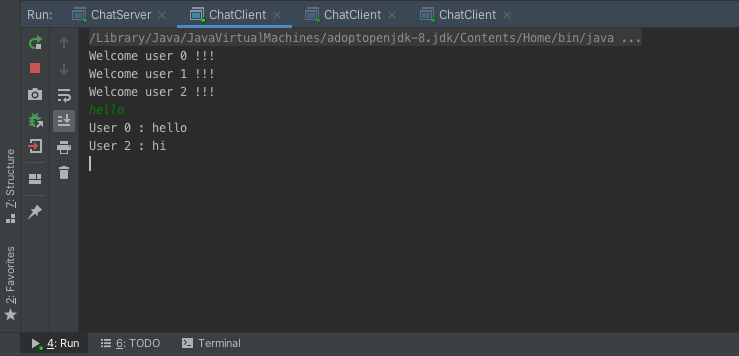
客户端1发送消息

客户端2接收消息

客户端3接收消息

客户端3发送信息



客户端2接受消息

客户端1接收消息

## 1.5 总结与心得

这回在做Socket大作业之前，很少真正的写过Java代码，最多写了些实验报告，只是知道些大体理论上的知识，但是真正写的时候才将自己整个的知识体系串联起来。

通过这次大作业，我学到许多，了解到了更多的偏向底层的东西，比如Socket和Thread，

因为自己有自己的兴趣，平时是写前端的，写的东西在浏览器上跑，浏览器上的调度模型是事件循环，写的代码全要听浏览器的话，根本接触不到线程这种触及到OS层面的东西。浏览器中的通信模型是http，最多在由http升级成websocket，只能接触到OSI七层网络模型中的应用层，通过写Socket终于接触到偏向于传输层附近的东西（Socket不算协议，算接口，所以说“附近”）。

这次Java大作业一确实让我学到了许多，也让我认识到了更多，让我有了许多技术态度方面的想法。

1.对于技术不能畏惧。软件行业是一个发展很快的行业，每过一段时间就会出一个新的技术是很正常的事，对于这些新技术不能畏惧，因为这些新技术都是基于计算机基础的，只要把基础巩固好就自然能知晓新技术的原理；

2.对于技术不能局限于自己的方向。要扩宽自己的视野，技术往往是相同的，学会一个方向如果不会其他涉及到的方面，必定会被自己的宽度所限制，比如前端开发人员如果不会后端，必定难以接触到更多的业务需求，被自己的宽度限制。