多客户端聊天服务器

实现要求:

- 1. 能同时接受多个客户端的连接
- 2. 将任意一个客户端发送的文本向所有客户端(包括发送方)转发

实现细节:

启动服务器

```
public static void main(String[] args) throws IOException{
    // 建立服务端的socket
   ServerSocket s = new ServerSocket(PORT);
    System.out.println("Sever Starting...");
    try {
       while (true) {
           // 等待客户端连接
           Socket socket = s.accept();
           System.out.println("Connect to Client : ");
           try {
               // 建立一个线程去处理该客户端的请求
               JabberServer clientThread = new JabberServer(socket);
               // 运行该线程
               clientThread.start();
               // 加入到客户端队列
               listClient.add(clientThread);
           }catch (IOException e) {
               socket.close();
           }
       }
    finally{
       s.close();
    }
```

对接单个客户端的线程:

```
static class JabberServer extends Thread {
    private Socket socket;
    private BufferedReader in;
    private PrintWriter out;
    public String username;
    Server server = new Server();

    public JabberServer(Socket s) throws IOException{
        socket = s;
        in = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
        out = new PrintWriter(new BufferedWriter(new
OutputStreamWriter(socket.getOutputStream())),true);
    }

    public void run(){
        ...
    }
}
```

客户端连接上服务器之后的提示语:

其中JabberServer的run函数会等待用户输入,然后将消息打印在服务器、广播给其他客户端,**最后打印在自己的客户端**。这样的消息显示循序在现实应用中十分重要,因为聊天消息一般先发送给其他客户端,等到其他客户端都收到之后,才在自己的客户端显示自己的消息。

```
String line = in.readLine();

if(line.equals("END"))

break;

// 服务器显示信息

System.out.println(this.username + " says: " + line);

// 广播

for(JabberServer client : listClient){

if(client == this)

continue;

// 指定是this这个客户端

new SendMsg(line, client.socket, this).start();

}

// 在客户端显示
out.println("my message: " + line);

}
```

发送消息的线程:

```
static class SendMsg extends Thread {
       Socket socket;
       String msg;
       String username;
       private BufferedReader in;
       private PrintWriter out;
       public SendMsg(String msg, Socket socket, JabberServer client) throws
IOException {
            this.socket = socket;
            this.msq = msq;
            this.username = client.username;
            in = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
            out = new PrintWriter(new BufferedWriter(new
OutputStreamWriter(socket.getOutputStream())),true);
       public void run(){
            super.run();
           // 在各自客户端发送消息
           out.println(username +": " +msg);
    }
```

功能展示:

如何运行:

```
java -jar P3_chatting_bot.jar
```

```
(base) → P3_chatting_bot java -jar P3_chatting_bot.jar
Sever Starting...
```

如上图,服务器已经开启,服务器的端口号设置的是8080

开启客户端

首先用nc连接服务器,测试消息转发情况:

```
nc localhost 8080
```

```
(base) → ~ nc localhost 8080
Pleaning enter username:
```

会要求输入一个用户名(测试中输入t1作为用户名)

会有一句欢迎语句,然后就等待用户输入。

此时开启另一个客户端(取名为t2)

此时t1会收到t2开启的消息:

```
(base) → ~ nc localhost 8080
Pleaning enter username:
  t1
Hi t1! Let's start chatting :P
  *****************************

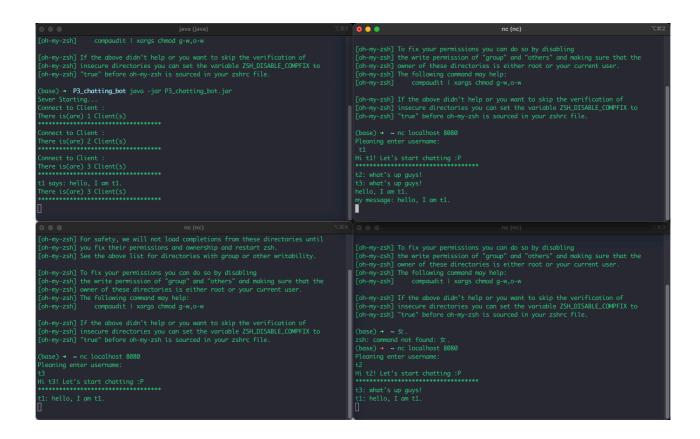
t2: what's up guys!
```

同时,服务器会有连接客户端数量的记录

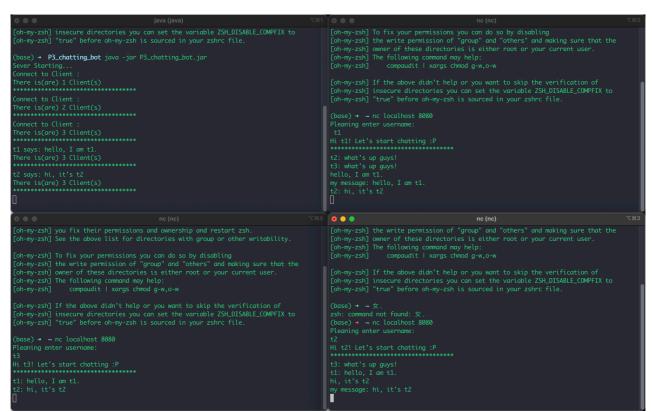
客户端之间发送消息

为验证多个客户端的效果,再开启一个客户端t3:

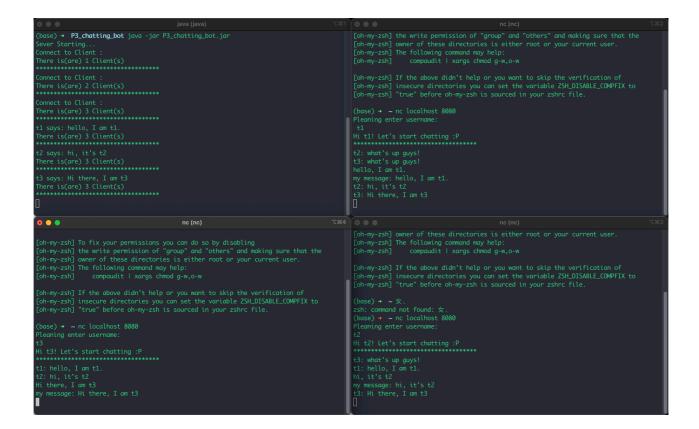
首先t1发送一则消息,服务器会显示t1发送的内容,t2和t3会收到t1以及它发送的消息。最后,t1自己会收到自己消息的提示语my message



同样的,t2发送消息。其他的客户端以及t2自己会收到消息。

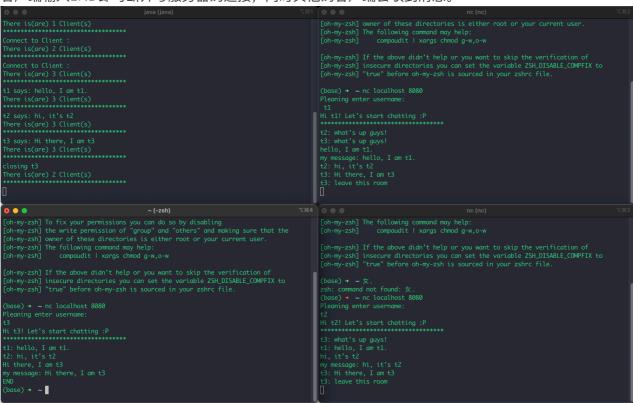


类似的, t3发送消息, 其他客户端和t3自己会收到消息。

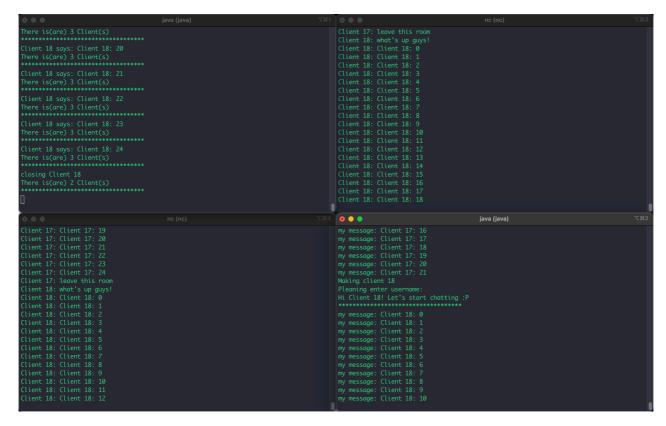


断开连接

客户端输入END即可断开与服务器的连接,同时其他的客户端会收到消息。



客户端测试代码(借用老师的课程示例代码)



发现服务端和客户端可以正常收发消息。