report

姓名: 巩德志 学号: 3200105088

服务器设计

运行server.jar后,输入需要监听的端口号,每当有一个客户端连接到服务器时,服务器就新开一个线程 讲行处理

```
server_socket = new ServerSocket(port);
while(true){
    Socket socket = server_socket.accept();
    socketList.add(socket); //把sock对象加入sock集合
    ServerBO_Thread st=new ServerBO_Thread(socket,
    socketList); //初始化多线程
    st.start();//启动多线程
```

ServerBO_Thread中最主要的方法是run,监听并读取客户端发来的消息,并通过遍历socket将读取的消息发送给所有客户端。如果有客户端退出,遍历socket将退出的消息发送给所有客户端,关闭线程即可。

```
while (true) {// 监视当前客户端有没有发来消息
   if (!client.isClosed()) {
       receive = bo.readDecrypt(input);
       clients.trimToSize();
       String[] param = receive.split(regex: "&");
       if (")start".equals(param[1])) { // 分析客户端发来的内容
           send = param[0] + "进入聊天室
           send = param[0] + "说: " + param[1];
       if (!("exit".equals(param[1]))) {// exit为退出聊天室信号
           for (Socket socket: clients) { // 遍历socket集合
              if (!socket.isClosed()) {
                  output = new DataOutputStream(socket.getOutputStream());
                  bo.encryptWrite(send, output);
       } else {// 如果有客户端退出
           for (Socket socket: clients) { // 遍历socke集合
                  if (!(socket.isClosed())) {
                     output = new DataOutputStream(socket.getOutputStream());
                      bo.encryptWrite(param[0] + "已退出聊天室", output);
           output = new DataOutputStream(client.getOutputStream());
           bo.encryptWrite(src: "exit", output);// 返回信号给要退出的客户端,然后关闭线程
           client.close();
           input.close();
          output.close();
       break;
```

实现了encryptWrite和readDecrypt来加密和读写数据流,加密方式为字符ASCII+13,2333在字符串末 尾用作结束符

```
public void encryptWrite(String src, DataOutputStream output) throws
IOException {
   char[] char_arr = src.toCharArray();
   // 加密操作
   for (int i = 0; i < char_arr.length; i++) {</pre>
        output.writeChar(char_arr[i] + 13);
   output.writeChar(v: 2333);
   output.flush();
// 读取并解密
public String readDecrypt(DataInputStream input) throws IOException {
    StringBuffer rtn = new StringBuffer(str: "");
   while (true) {
        int char src = input.readChar();
            rtn.append((char) (char src - 13));
            break;
    return rtn.toString();
```

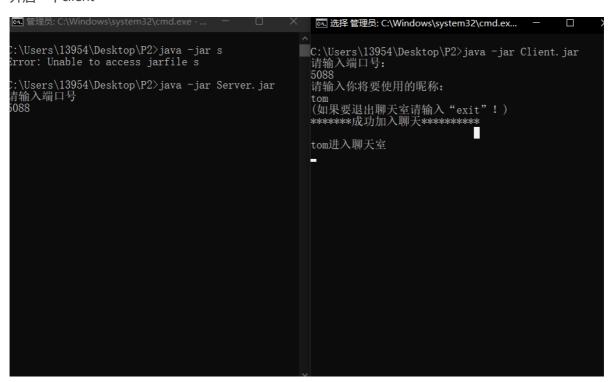
客户端设计

运行client.jar后,输入需要连接的端口号和用户名,运行线程收发数据

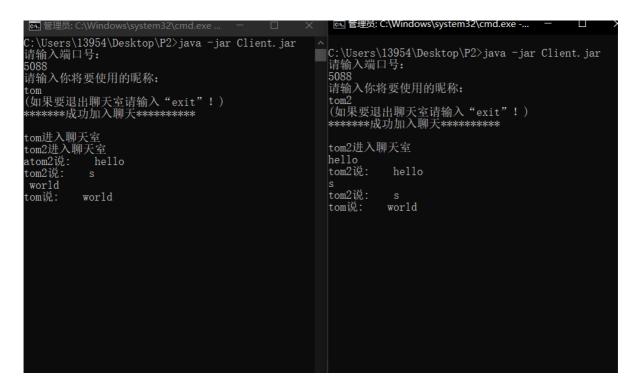
```
encryptWrite(send,output);
Out out=new Out(output,name,input,socket);
out.start();//启动发送聊天内容的多线程
while(true){
    String receive = readDecrypt(input);
    if("exit".equals(receive)){//如果收到"exit"则退出聊天室
        System.out.println(x: "************************);
    input.close();
    output.close();
    socket.close();
    System.exit(status: 0);
    }
    System.out.println(receive);
}
```

实验结果

开启一个client



开启两个client, 发送消息



用户退出聊天

