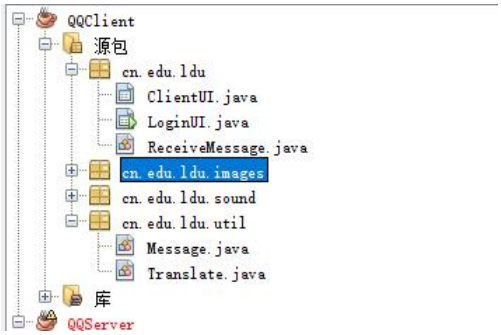
UDP 通信

实验介绍

本次实验是完成类似 QQ 群聊的设计，客户机与服务器采用 UDP 协议进行数据传输，服务器启动之后，客户机使用 DatagramSocket(int port,InetAddress laddr)定义一个 UDP 数据套接字，绑定到服务器的 ip 地址和端口，随后可以与服务器进行通信。

程序结构和功能



QQClinet 客户端函数和功能

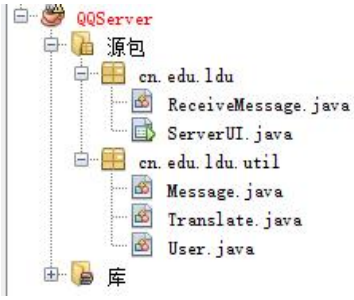
ClientUI.java:客户端会话界面，分为发送消息窗口和会话消息窗口、显示在线用户

LoginUI.java:客户登录验证界面

ReceiveMessage.java:客户机接收消息线程

Message.java:消息体类，定义会话消息结构

Translate.java:对象的序列化和反序列化



QQSever 服务端函数和功能

ServerUI.java:服务器 UI 界面

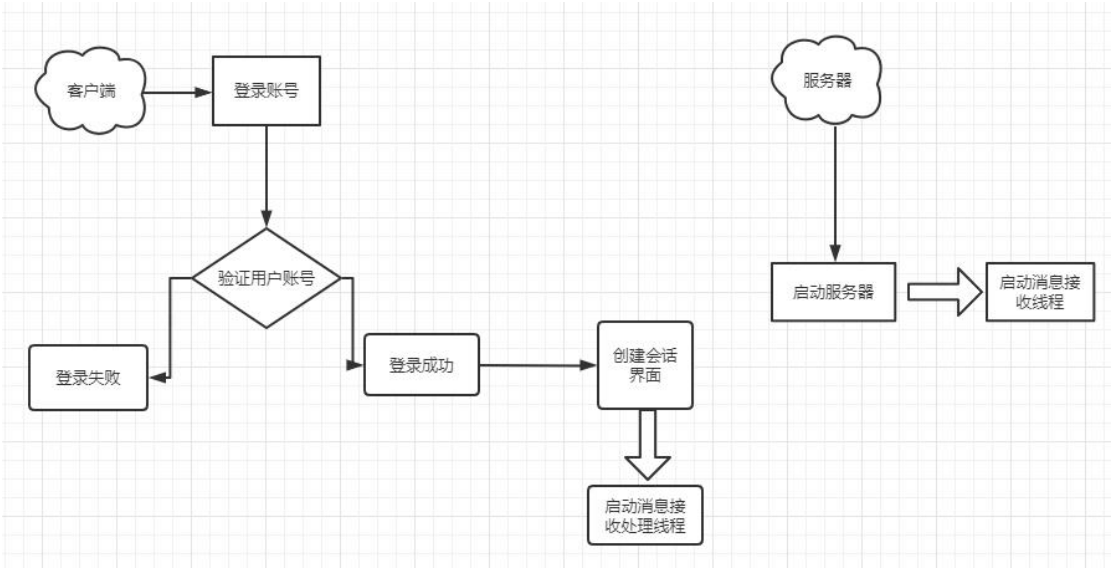
ReceiveMessage.java:服务器接收消息和处理消息的线程类

User.java:User 类，定义用户对象，包含用户名和收到的报文

Message.java:消息体类，定义会话消息结构

Translate.java:对象的序列化和反序列化

程序流程图



消息协议设计

在客户端和服务器中存在多种不同类型的消息类型，对于不同的消息类型，服务器应该给出不同的处理方式，而且还需要规定相应的协议和规范。比如服务器收到的消息类型为“登录消息”，服务器应该验证客户的登录信息；收到的消息类型为“普通消息文本”，服务器应该向其余在线用户转发此信息，显示在他们的消息面板中。

客户机与服务器之间的消息类型

M_LOGIN	client 向 server 登录时发送 M_LOGIN 消息；client 登录成功后 server 向其他所有在线 client 转发该用户的 M_LOGIN 消息，目的是让其他用户加入各自的在线列表中(提醒所有用户有新的用户登录在线)
M_SUCCESS	client 登录成功后 server 向 client 回送 M_SUCCESS 消息，弹出新的对话窗口，同时关闭登录窗口
M_FAILURE	client 登录失败后 server 向 client 回送 M_FAILURE 消息，提醒 client 登录失败
M_ACK	client 登录成功后，server 向登录用户回送应答消息 M_ACK,将其他在线用户的 id 发送给登录用户
M_MSG	client 向 server 发送普通文本消息时，server 向所有 client 转发 M_MSG
M_QUIT	当 client 断开服务器连接时,触发 M_QUIT，server 向所有其他用户转发 M_QUIT 消息

部分关键代码

```
//启动服务器
private void btnStartActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        //获取服务器工作地址端口
        String hostName=txtHostName.getText();
        int hostPort=Integer.parseInt(txtHostPort.getText());
        //创建UDP数据报套接字,在指定端口侦听
        DatagramSocket serverSocket=new DatagramSocket(hostPort);
        txtArea.append("服务器开始侦听...\n");
        //创建并启动UDP消息接收线程
        Thread recvThread=new ReceiveMessage(serverSocket,this); // 把当前的ServerUI类当作参数传过去
        recvThread.start();
```

服务器启动事件，启动端口在指定端口侦听，这里另外开辟了一个 UDP 消息接收线程，通过消息线程封装处理来自客户端的消息。

```
public void run() {
    while (true) { //循环处理收到的各种消息
        try {
        packet=new DatagramPacket(data,data.length);//构建接收报文,接收到的数据存放到data
        serverSocket.receive(packet);//接收客户机数据,接收到的数据存放到data
        //收到的数据转为消息对象
        Message msg=(Message)Translate.ByteToObject(packet.getData());
        String userId=msg.getUserId();//当前消息来自用户的id
        if (msg.getType().equalsIgnoreCase("M_LOGIN")) { //是M_LOGIN消息
            Message backMsg=new Message();
            //假定只有2000、3000、8000三个帐号可以登录
            if (!userId.equals("2000") && !userId.equals("3000") && !userId.equals("8000")) {//登录不成功
                backMsg.setType("M_FAILURE");
                byte[] buf=Translate.ObjectToByte(backMsg);
                DatagramPacket backPacket=new DatagramPacket(buf,buf.length,packet.getAddress(),packet.getPort());//向登录用户发
                serverSocket.send(backPacket); //发送
            }else { //登录成功
                backMsg.setType("M_SUCCESS");
                byte[] buf=Translate.ObjectToByte(backMsg);
                DatagramPacket backPacket=new DatagramPacket(buf,buf.length,packet.getAddress(),packet.getPort());//向登录用户发
                serverSocket.send(backPacket); //发送

                User user=new User();
                user.setUserId(userId); //用户名
                user.setPacket(packet); //保存收到的报文
                userList.add(user); //将新用户加入用户列表

                //更新服务器聊天室大厅
                parentUI.txtArea.append(userId+" 登录! \n");
```

```
//向所有其他在线用户发送M_LOGIN消息，向新登录者发送整个用户列表
for (int i=0;i<userList.size();i++) { //遍历整个用户列表
    //向其他在线用户发送M_LOGIN消息
    if (!userId.equalsIgnoreCase(userList.get(i).getUserId())){
        DatagramPacket oldPacket=userList.get(i).getPacket();
        DatagramPacket newPacket=new DatagramPacket(data,data.length,oldPacket.getAddress(),oldPacket.getPort
        serverSocket.send(newPacket); //发送
    }//end if
    //向当前用户回送M_ACK消息，将第i个用户加入当前用户的用户列表
    // 将其他在线用户名单传递给当前登录成功的用户
    Message other=new Message();
    other.setUserId(userList.get(i).getUserId()); // 设置传送消息中遍历到的userId
    other.setType("M_ACK");
    byte[] buffer=Translate.ObjectToByte(other);
    DatagramPacket newPacket=new DatagramPacket(buffer,buffer.length,packet.getAddress(),packet.getPort());
    serverSocket.send(newPacket);
}//end for
```

在服务器接收线程的 run 方法里面，分别对不同消息类型进行相应的处理。在其中的 for 循环里面 serverSocket.send(new Packet)给其他用户转发登录成功的用户的 M_LOGIN 消息，向当前用户转发 M_ACK 消息，把其他在线用户的名单传递给当钱登录成功的用户。

在客户端的“登录按钮”中绑定登录事件，在登录事件中获取服务器地址和 ip 端口，并创建 UDP 套接字，构建当前用户的 Message 消息体，将消息对象实例化为 byte 传递给服务器。

```
Message msg=new Message();
msg.setUserId(id);//登录名
msg.setPassword(password); //密码
msg.setType("M_LOGIN"); //登录消息类型
msg.setToAddr(remoteAddr); //目标地址
msg.setToPort(remotePort); //目标端口
byte[] data=Translate.ObjectToByte(msg); //消息对象序列化
//定义登录报文
DatagramPacket packet=new DatagramPacket(data,data.length,remoteAddr,remotePort);
//发送登录报文
clientSocket.send(packet);
```

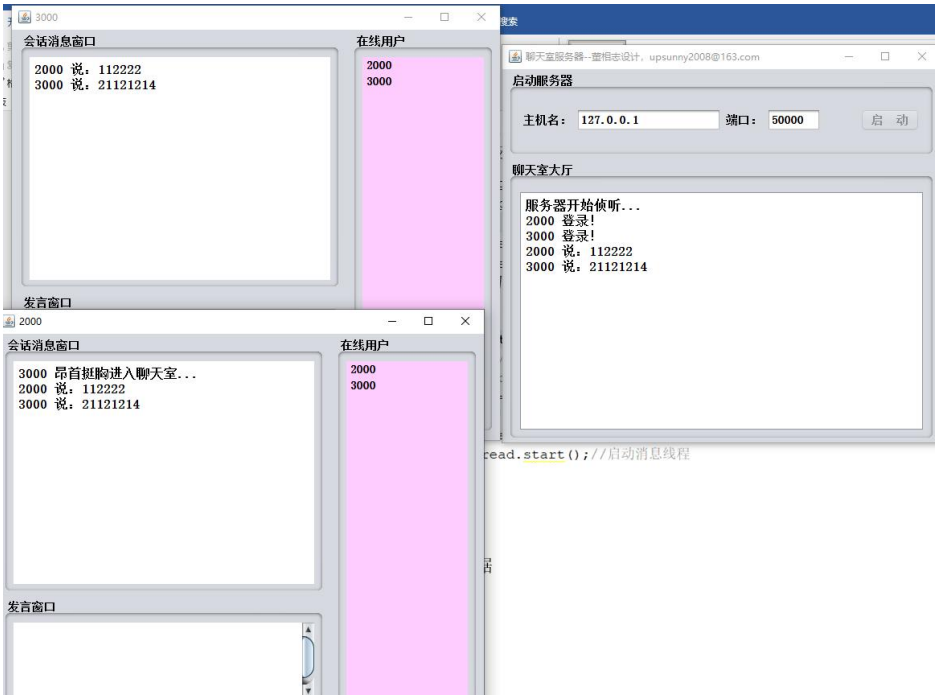
登录成功后关闭当前页面，创建客户机界面

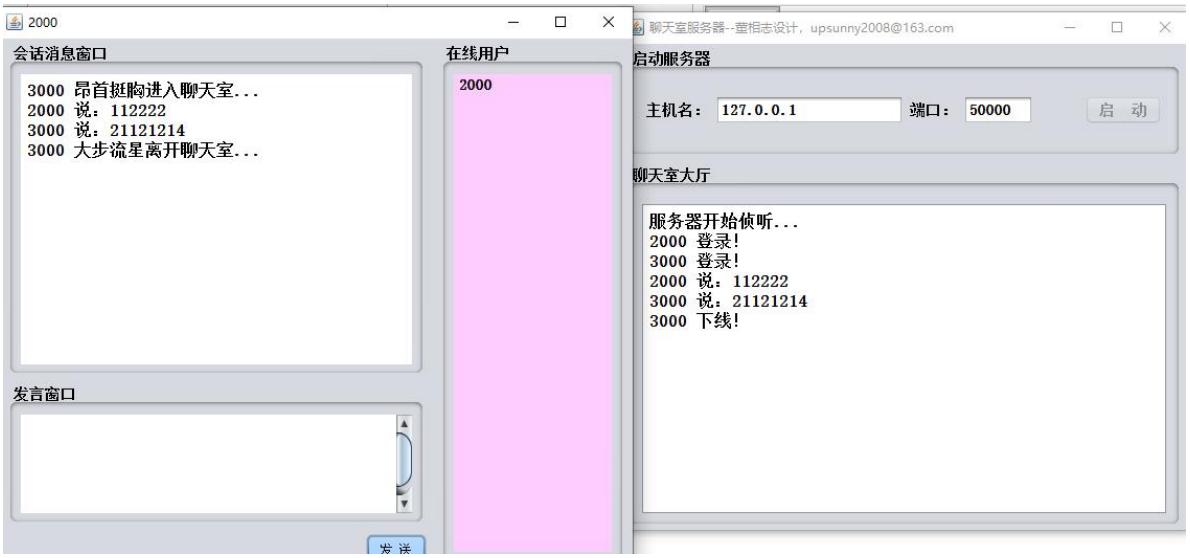
```
if (backMsg.getType().equalsIgnoreCase("M_SUCCESS")) { //登录成功
    this.dispose(); //关闭登录对话框
    ClientUI client=new ClientUI(clientSocket,msg); //创建客户机界面
    client.setTitle(msg.getUserId()); //设置标题
    client.setVisible(true); //显示会话窗体
```

在 ClientUI 初始化构造函数中随之创建了消息接收和处理线程，对不同类型消息类型接收和处理。

```
public ClientUI(DatagramSocket socket,Message msg) {
    this(); //调用无参数构造函数，初始化界面
    clientSocket=socket; //初始化会话套接字
    this.msg=msg; //登录消息
    //创建客户机消息接收和处理线程
    Thread recvThread=new ReceiveMessage(clientSocket,this);
    recvThread.start();//启动消息线程
}
```

实验运行数据


3000
会话消息窗口
2000 说：112222
3000 说：21121214
在线用户
2000
3000
发言窗口
聊天室服务器--董相志设计，upsunny2008@163.com
启动服务器
主机名：
127.0.0.1
端口：
50000
启 动
聊天室大厅
服务器开始侦听...
2000 登录!
3000 登录!
2000 说：112222
3000 说：21121214
2000
会话消息窗口
3000 昂首挺胸进入聊天室...
2000 说：112222
3000 说：21121214
在线用户
2000
3000
发言窗口
ead.start();//启动消息线程


2000
会话消息窗口
3000 昂首挺胸进入聊天室...
2000 说：112222
3000 说：21121214
3000 大步流星离开聊天室...
在线用户
2000
发言窗口
聊天室服务器--董相志设计，upsunny2008@163.com
启动服务器
主机名：
127.0.0.1
端口：
50000
启 动
聊天室大厅
服务器开始侦听...
2000 登录!
3000 登录!
2000 说：112222
3000 说：21121214
3000 下线!