



廣東工業大學

课 程 设 计

课程名称	<u>计算机网络</u>
题目名称	<u>基于 Socket 的即时通信系统</u>
学生学院	<u>计算机学院</u>
专业班级	<u>计算机科学与技术 1 班</u>
学 号	<u>3119004760</u>
学生姓名	<u>叶嘉轩</u>
指导教师	<u>彭重嘉</u>

2021 年 7 月 3 日

目录

1.项目要求.....	2
1. 编程语言: java.....	2
2. 开发工具: IDEA.....	2
3. 运行平台: macOS.....	2
4.项目设计.....	3
5.测试.....	10
6.开发计划.....	16
7.总结.....	16
8.应收集的资料及主要参考文献.....	16

1.项目要求

- 1.利用 Socket 通信机制实现一种即时通信系统。
- 2.系统分为服务端和客户端两部分:
 - 2.1 客户端向服务器注册自己的信息, 包括 IP 地址、端口号、用户名等;
 - 2.2 客户端启动后向服务器查询其它用户在线状态, 包括用户名、IP 地址、端口号等;
 - 2.3 客户端之间建立直接连接, 实现一对一的信息交换;
 - 2.4 服务器端维护用户的状态信息, 对用户查询做出响应;
 - 2.5 利用客户端, 或是服务器实现信息的群发(实现时任选一种方式)。
- 3.系统必须对出现的问题或错误做出响应

2.需求分析

利用 Socket API 通信机制实现一种即时通信系统; 系统分为服务端和客户端两部分。主要目标:

- 1) 客户端能向服务端发送包括: 群发业务、私聊业务
- 2) 客户端能接收并处理发送给自己的消息和业务
- 3) 客户端连接服务端时需注册自己的用户名
- 4) 服务端能接收并处理客户端的业务请求
- 5) 当有客户端连接上服务端或者有客户端退出连接时, 服务端得广播当前在线用户。

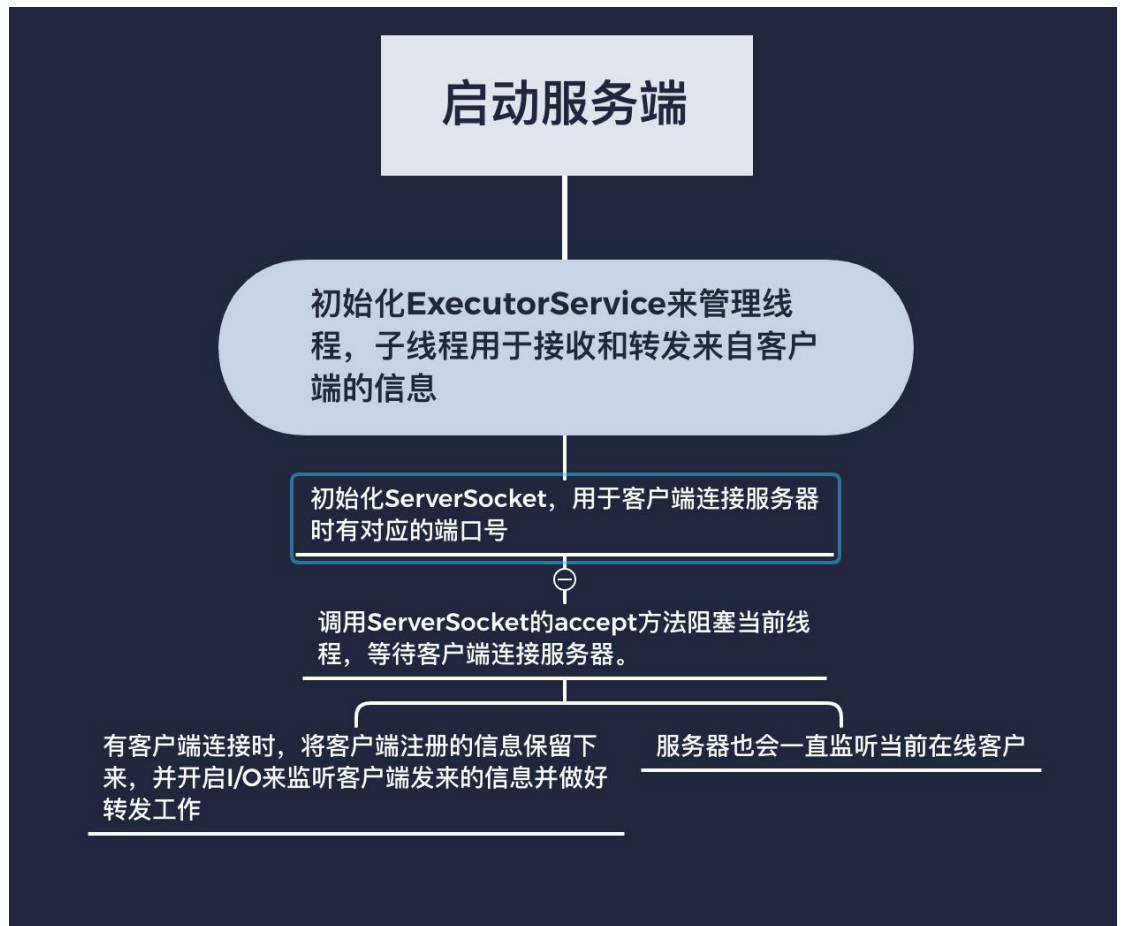
3.开发环境

1. 编程语言: java
2. 开发工具: IDEA
3. 运行平台: macOS

4.项目设计

1. 总体设计

1) 服务端功能结构图:



2) 客户端功能结构图:



2. 封装的信息类型

通过信息的 TYPE 来判断服务端当前接收到来自客户端发出的信息是登录、登出还是私聊群发信息。

关键代码：

```
public class MessageType {
    public static final int TYPE_LOGIN = 0x1; //登录消息类型
    public static final int TYPE_SEND_SINGLE = 0x2; //私聊信息
    public static final int TYPE_SEND_GROUP = 0x3; //群发信息
    public static final int TYPE_LOGIN_OUT = 0x4; //登出消息类型
}
```

3. 封装的信息 MessageInfo

客户端发送的信息和服务端转发的信息都是经过包装的 MessageInfo

关键代码：

import java.io.Serializable;

```
public class MessageInfo implements Serializable {
    private String from; //发送者
    private String to; //接收者
    private int type; //消息类型
    private String info; //消息内容

    public MessageInfo() {

    }
}
```

```
public MessageInfo(String from, String to, int type, String info) {  
    this.from = from;  
    this.to = to;  
    this.type = type;  
    this.info = info;  
}
```

```
public String MessageInfo() {  
    return "Message{" +  
        "from='" + from + "\" +  
        ", to='" + to + "\" +  
        ", type=" + type +  
        ", info='" + info + "\" +  
        '}'";  
}
```

```
public String getFrom() {  
    return from;  
}
```

```
public void setFrom(String from) {  
    this.from = from;  
}
```

```
public String getTo() {  
    return to;  
}
```

```
public void setTo(String to) {  
    this.to = to;  
}
```

```
public int getType() {  
    return type;  
}
```

```
public void setType(int type) {  
    this.type = type;  
}
```

```
public String getInfo() {  
    return info;  
}
```

```

    public void setInfo(String info) {
        this.info = info;
    }
}

```

4. 客户端业务处理函数设计

① 客户端接收来自服务端的注册信息:

- 1.通过 `socket.getOutputStream()`和 `socket.getInputStream()`客户端会收到来自服务端发来的信息（注册用户名的信息）。
- 2.客户端注册的用户名会被打包成 `MessageInfo`，然后通过输出流传输给了服务端。
- 3.然后客户端就会启动子线程去接收来自服务端转发来的信息，以及去转发客户端要传送的信息。

关键代码如下:

```

Socket socket = new Socket("localhost",9988); //Localhost 代表的是本地的服务器，要是连别的服务器，就带 IP 地址
System.out.println("连接服务器成功");
ObjectOutputStream outputStream = new
ObjectOutputStream(socket.getOutputStream());
ObjectInputStream inputStream = new
ObjectInputStream(socket.getInputStream());
//客户端登录
System.out.println("请输入昵称: ");
String name = input.nextLine();

//对输入的信息进行封装，然后发给服务器
MessageInfo messageInfo = new
MessageInfo(name,null,MessageType.TYPE_LOGIN,null);
    outputStream.writeObject(messageInfo);

//从服务器读取信息
messageInfo = (MessageInfo) inputStream.readObject();
System.out.println(messageInfo.getInfo());
//登录结束

```

② 客户端会一直监听自己是否要传送信息出去:

- 1.用户会在客户端文件运行界面按照提示语发送信息。
- 2.输入好的信息同样会被打包成 `MessageInfo`，通过输出流发送出去。

关键代码:

```

//在主线程发送信息
while (true) {

```

```

        messageInfo = new MessageInfo();
        System.out.println("to:");
        messageInfo.setTo(input.nextLine());
        System.out.println("Info:");
        messageInfo.setInfo(input.nextLine());
        messageInfo.setFrom(name);
        if (messageInfo.getTo().equals("all")) {

            messageInfo.setType(MessageType.TYPE_SEND_GROUP);
        } else {

            messageInfo.setType(MessageType.TYPE_SEND_SINGLE);
        }
        outputStream.writeObject(messageInfo);
    }

```

- ③ 客户端会开辟一个子线程，子线程一直在等待服务端转发来的信息

关键代码：

```

private ObjectInputStream inStream;
private boolean flag = true;

public ReadInfoThread(ObjectInputStream inStream) {
    this.inStream = inStream;
}

public void setFlag(boolean flag) {
    this.flag = flag;
}

@Override
public void run() {
    //利用 flag 来判断是否一直开辟线程读取信息
    while(flag) {
        try {
            MessageInfo info = (MessageInfo) inStream.readObject();
            if (info.getType() == MessageType.TYPE_SEND_SINGLE)
            {
                System.out.println "[" + info.getFrom() + "] 对我说 : " +
info.getInfo());
            }else {
                System.out.println "[" + info.getFrom() + "] 广播 : " +
info.getInfo());
            }
        } catch (IOException e) {

```

```

        e.printStackTrace();
    } catch (ClassNotFoundException e) {
        e.printStackTrace();
    }
}
//flag 为 false 就会执行下面的代码
if (inStream != null) {
    try {
        inStream.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
}
}

```

5. 服务端业务处理函数设计

①阻塞当前线程，等待客户端连接

关键代码：

```

ServerSocket server = new ServerSocket(9988);
System.out.println("服务器启动，等待客户端连接");
while(true) {
    Socket socket = server.accept();//等待客户端连接，此方法会阻塞当前
    线程，直到有客户端连接
    UserThread user = new UserThread(socket,vector);
    s.execute(user);
}

```

②处理客户端传来的信息

1.对于私聊信息 TYPE_SEND_SINGLE，就根据用户名去转发给对应的客户端。

关键代码：

```

case MessageType.TYPE_SEND_SINGLE:
    String to = message.getTo();
    String from = message.getFrom();
    size = vector.size();
    for (int i=0; i < size; i++) {
        user = vector.get(i);
        if (from.equals(user.name)) {
            fromUser = vector.get(i);
        }
        if (to.equals(user.name) && user != this) {
            user.outputStrem.writeObject(message);
            flag = true;
            break;
        }
    }
}
}

```



```

if (flag == false) {
    message.setInfo("该客户端不在聊天室内");
    fromUser.outputStrem.writeObject(message);
    flag = true;
}
break;

```

2.对于群发信息 TYPE_SEND_GROUP, 服务端就会向所有客户端转发此消息

关键代码:

```

case MessageType.TYPE_SEND_GROUP:
    size = vector.size();
    for (int i=0; i<size; i++) {
        user = vector.get(i);
        if (user != this) {
            user.outputStrem.writeObject(message);
        }
    }
break;

```

3.对于登录信息, 就广播当前的在线客户

关键代码:

```

case MessageType.TYPE_LOGIN:
    name = message.getFrom();
    getUserInfo(this.vector, message,
        MessageType.TYPE_LOGIN);
    break;

```

③服务端要时刻监听当前聊天室在线的用户

1.通过在有新客户端连接以及有客户端退出时, 广播当前仍在线的客户端

关键代码:

```

String userGroup = "当前在线客户端有: ";
UserThread user;
int size = users.size();
for (int i=0; i<size; i++) {
    user = users.get(i);
    userGroup = userGroup + " " + user.name;
}
System.out.println(users);
System.out.println(userGroup);
if (type == MessageType.TYPE_LOGIN) {
    message.setInfo("欢迎你加入聊天室" + userGroup);
    try {
        this.outputStrem.writeObject(message);
    } catch (IOException e) {
        e.printStackTrace();
    }
}

```

```

    }
    message.setFrom(name);
    message.setTo("all");
    message.setType(MessageType.TYPE_SEND_GROUP);
    message.setInfo(name + "加入聊天室" + "," + userGroup);
    for (int i=0; i<size; i++) {
        try {
            user = users.get(i);
            if (user != this) {
                user.outputStrem.writeObject(message);
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
} else {
    //登出信息
    message.setTo("all");
    message.setInfo(message.getFrom() + "退出聊天室" + "," +
userGroup);
    message.setType(MessageType.TYPE_SEND_GROUP);
    for (int i=0; i<size; i++) {
        user = vector.get(i);
        try {
            user.outputStrem.writeObject(message);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
}
}
}

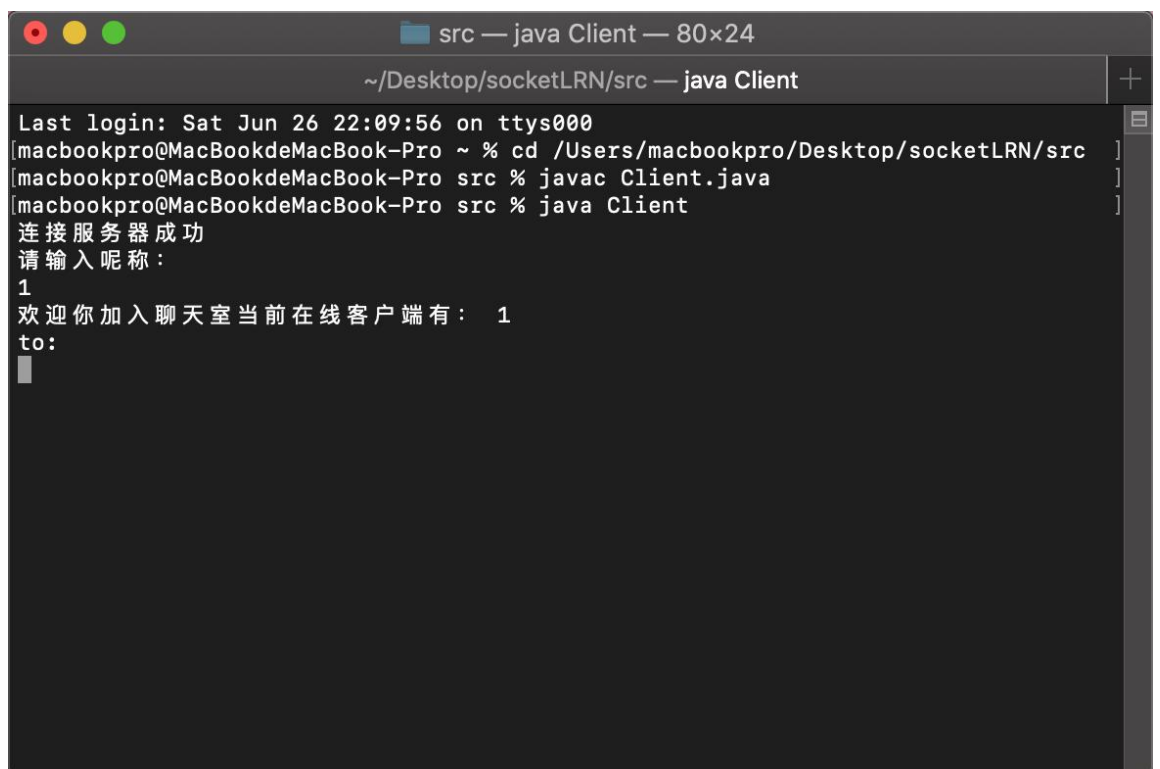
```

5.测试

1. 打开服务器测试：服务器启动等待客户端连接



打开客户端连接测试: 客户端是直接根据 localhost 以及端口号连接到本主机的服务器的。



当有多台客户端连接到服务端时:



```
src — java Client — 80x24
~/Desktop/socketLRN/src — java Client
Last login: Sat Jun 26 22:09:56 on ttys000
macbookpro@MacBookdeMacBook-Pro ~ % cd /Users/macbookpro/Desktop/socketLRN/src ]
macbookpro@MacBookdeMacBook-Pro src % javac Client.java ]
macbookpro@MacBookdeMacBook-Pro src % java Client ]
连接服务器成功
请输入昵称:
1
欢迎你加入聊天室当前在线客户端有: 1
to:
[2] 广播 : 2加入聊天室,当前在线客户端有: 1 2
[]

src — java Client — 80x23
~/Desktop/socketLRN/src — java Client
Last login: Sun Jul 4 00:26:16 on ttys000
macbookpro@MacBookdeMacBook-Pro ~ % cd /Users/macbookpro/Desktop/socketLRN/src ]
macbookpro@MacBookdeMacBook-Pro src % javac Client.java ]
macbookpro@MacBookdeMacBook-Pro src % java Client ]
连接服务器成功
请输入昵称:
2
欢迎你加入聊天室当前在线客户端有: 1 2
to:
[]
```

```
src — java Client — 80x24
~/Desktop/socketLRN/src — java Client
Last login: Sat Jun 26 22:09:56 on ttys000
macbookpro@MacBookdeMacBook-Pro ~ % cd /Users/macbookpro/Desktop/socketLRN/src ]
macbookpro@MacBookdeMacBook-Pro src % javac Client.java ]
macbookpro@MacBookdeMacBook-Pro src % java Client ]
连接服务器成功
请输入昵称：
1
欢迎你加入聊天室当前在线客户端有： 1
to:
[2] 广播：2加入聊天室,当前在线客户端有： 1 2
[3] 广播：3加入聊天室,当前在线客户端有： 1 2 3
[]

src — java Client — 80x23
~/Desktop/socketLRN/src — java Client
Last login: Sun Jul 4 00:26:16 on ttys000
macbookpro@MacBookdeMacBook-Pro ~ % cd /Users/macbookpro/Desktop/socketLRN/src ]
macbookpro@MacBookdeMacBook-Pro src % javac Client.java ]
macbookpro@MacBookdeMacBook-Pro src % java Client ]
连接服务器成功
请输入昵称：
2
欢迎你加入聊天室当前在线客户端有： 1 2
to:
[3] 广播：3加入聊天室,当前在线客户端有： 1 2 3
[]

src — java Client — 80x22
~/Desktop/socketLRN/src — java Client
Last login: Sun Jul 4 00:26:18 on ttys002
macbookpro@MacBookdeMacBook-Pro src % cd /Users/macbookpro/Desktop/socketLRN/src ]
macbookpro@MacBookdeMacBook-Pro src % javac Client.java ]
macbookpro@MacBookdeMacBook-Pro src % java Client ]
连接服务器成功
请输入昵称：
欢迎你加入聊天室当前在线客户端有： 1 2 3
to:
[ ]
```

2. 群发消息测试

客户端在发送信息时，to 的接收者为 all 时就被设定为群发信息

```
src — java Client — 80x24
~/Desktop/socketLRN/src — java Client
Last login: Sat Jun 26 22:09:56 on ttys000
macbookpro@MacBookdeMacBook-Pro ~ % cd /Users/macbookpro/Desktop/socketLRN/src
macbookpro@MacBookdeMacBook-Pro src % javac Client.java
macbookpro@MacBookdeMacBook-Pro src % java Client
连接服务器成功
请输入昵称:
1
欢迎你加入聊天室当前在线客户端有: 1
to:
[2] 广播: 2加入聊天室,当前在线客户端有: 1 2
[3] 广播: 3加入聊天室,当前在线客户端有: 1 2 3
[2] 广播: Hello!
]

src — java Client — 80x21
~/Desktop/socketLRN/src — java Client
Last login: Sun Jul 4 00:26:18 on ttys002
macbookpro@MacBookdeMacBook-Pro src % cd /Users/macbookpro/Desktop/socketLRN/src
macbookpro@MacBookdeMacBook-Pro src % javac Client.java
macbookpro@MacBookdeMacBook-Pro src % java Client
连接服务器成功
请输入昵称:
3
欢迎你加入聊天室当前在线客户端有: 1 2 3
to:
Info:
[2] 广播: Hello!
]

src — java Client — 80x23
~/Desktop/socketLRN/src — java Client
Last login: Sun Jul 4 00:26:16 on ttys000
macbookpro@MacBookdeMacBook-Pro ~ % cd /Users/macbookpro/Desktop/socketLRN/src
macbookpro@MacBookdeMacBook-Pro src % javac Client.java
macbookpro@MacBookdeMacBook-Pro src % java Client
连接服务器成功
请输入昵称:
2
欢迎你加入聊天室当前在线客户端有: 1 2
to:
[3] 广播: 3加入聊天室,当前在线客户端有: 1 2 3
all
Info:
Hello!
to:
[1] 广播: Hello!
]
```

3. 私聊消息测试

用户 1 单独给用户 2 发出一条信息，这是可以观察到用户 2 是可以接收到 1 发来的信息的。

```
src — java Client — 80x24
~/Desktop/socketLRN/src — java Client
Last login: Sat Jun 26 22:09:56 on ttys000
macbookpro@MacBookdeMacBook-Pro ~ % cd /Users/macbookpro/Desktop/socketLRN/src
macbookpro@MacBookdeMacBook-Pro src % javac Client.java
macbookpro@MacBookdeMacBook-Pro src % java Client
连接服务器成功
请输入昵称:
1
欢迎你加入聊天室当前在线客户端有: 1
to:
[2] 广播: 2加入聊天室,当前在线客户端有: 1 2
[3] 广播: 3加入聊天室,当前在线客户端有: 1 2 3
[2] 广播: Hello!
2
Info:
我是1
to:
[1] 广播: Hello!
]

src — java Client — 80x23
~/Desktop/socketLRN/src — java Client
Last login: Sun Jul 4 00:26:16 on ttys000
macbookpro@MacBookdeMacBook-Pro ~ % cd /Users/macbookpro/Desktop/socketLRN/src
macbookpro@MacBookdeMacBook-Pro src % javac Client.java
macbookpro@MacBookdeMacBook-Pro src % java Client
连接服务器成功
请输入昵称:
2
欢迎你加入聊天室当前在线客户端有: 1 2
to:
[3] 广播: 3加入聊天室,当前在线客户端有: 1 2 3
all
Info:
Hello!
to:
[1] 广播: Hello!
]
```

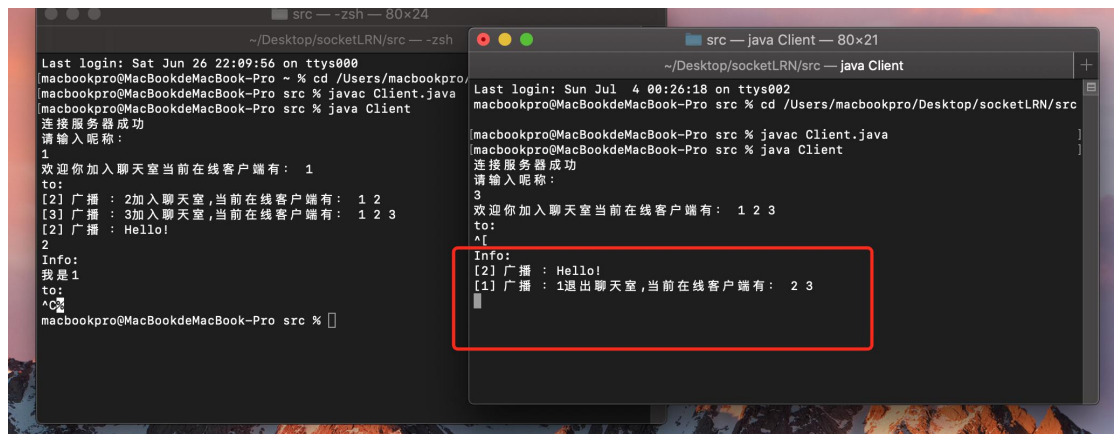
但是用户3是收不到来自用户1发给用户2的信息的

```
src — java Client — 80x21
~/Desktop/socketLRN/src — java Client
Last login: Sun Jul  4 00:26:18 on ttys002
macbookpro@MacBookdeMacBook-Pro src % cd /Users/macbookpro/Desktop/socketLRN/src
macbookpro@MacBookdeMacBook-Pro src % javac Client.java
macbookpro@MacBookdeMacBook-Pro src % java Client
连接服务器成功
请输入昵称：
3
欢迎你加入聊天室当前在线客户端有： 1 2 3
to:
^[
Info:
[2] 广播 : Hello!
```

4.查看在线客户端测试

有客户端连接服务端或者退出去，服务端都会广播当前在线客户信息

```
src — java Client — 80x24
~/Desktop/socketLRN/src — java Client
Last login: Sat Jun 26 22:09:56 on ttys000
[macbookpro@MacBookdeMacBook-Pro ~ % cd /Users/macbookpro/Desktop/socketLRN/src
[macbookpro@MacBookdeMacBook-Pro src % javac Client.java
[macbookpro@MacBookdeMacBook-Pro src % java Client
连接服务器成功
请输入昵称：
1
欢迎你加入聊天室当前在线客户端有： 1
to:
[2] 广播 : 2加入聊天室,当前在线客户端有： 1 2
[3] 广播 : 3加入聊天室,当前在线客户端有： 1 2 3
[2] 广播 : Hello!
2
Info:
我是1
to:
```



```
src — zsh — 80x24
~/Desktop/socketLRN/src — zsh
Last login: Sat Jun 26 22:09:56 on ttys000
macbookpro@MacBookdeMacBook-Pro ~ % cd /Users/macbookpro/
macbookpro@MacBookdeMacBook-Pro src % javac Client.java
macbookpro@MacBookdeMacBook-Pro src % java Client
连接服务器成功
请输入昵称:
1
欢迎你加入聊天室当前在线客户端有: 1
to:
[2] 广播: 2加入聊天室,当前在线客户端有: 1 2
[3] 广播: 3加入聊天室,当前在线客户端有: 1 2 3
[2] 广播: Hello!
2
Info:
我是1
to:
^C
macbookpro@MacBookdeMacBook-Pro src %

src — java Client — 80x21
~/Desktop/socketLRN/src — java Client
Last login: Sun Jul 4 00:26:18 on ttys002
macbookpro@MacBookdeMacBook-Pro src % cd /Users/macbookpro/Desktop/socketLRN/src
macbookpro@MacBookdeMacBook-Pro src % javac Client.java
macbookpro@MacBookdeMacBook-Pro src % java Client
连接服务器成功
请输入昵称:
3
欢迎你加入聊天室当前在线客户端有: 1 2 3
to:
^[
Info:
[2] 广播: Hello!
[1] 广播: 1退出聊天室,当前在线客户端有: 2 3
```

6.开发计划

时间	完成内容
2021.6.13-2021.6.13	Socket api 的学习
2021.6.13-2021.6.14	程序设计
2021.6.15-2021.6.16	即时通信系统完成
2021.7.3-2021.7.4	课设报告

7.总结

整个项目实现了所有的项目要求，包括客户端向服务端注册自己的信息，实现了群发和私发信息。服务端会对当前在线用户进行一个及时的转发。

项目优点：功能完善。而且通信机制简单明了。

项目缺点：没有 UI 界面，输入输出都是在终端运行。

改进设想：每个客户端都有一个展示聊天界面的 UI。

8.应收集的资料及主要参考文献

《计算机网络》（第五版） 谢希仁