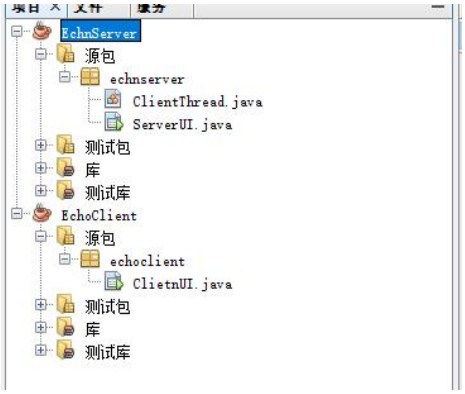
一客户一线程通信

实验介绍

本次实验是实现多个客户机可以连接一个服务器的功能，服务器对每一个新建立的客户机连接，都立即创建新线程与之一一对应，这种响应模式称为一客户一线程。不限制客户机数量，不同客户机可以同时向服务器发送任意字符串，服务器一对一原样回送，实现一客户一线程的工作模式。

程序结构和功能



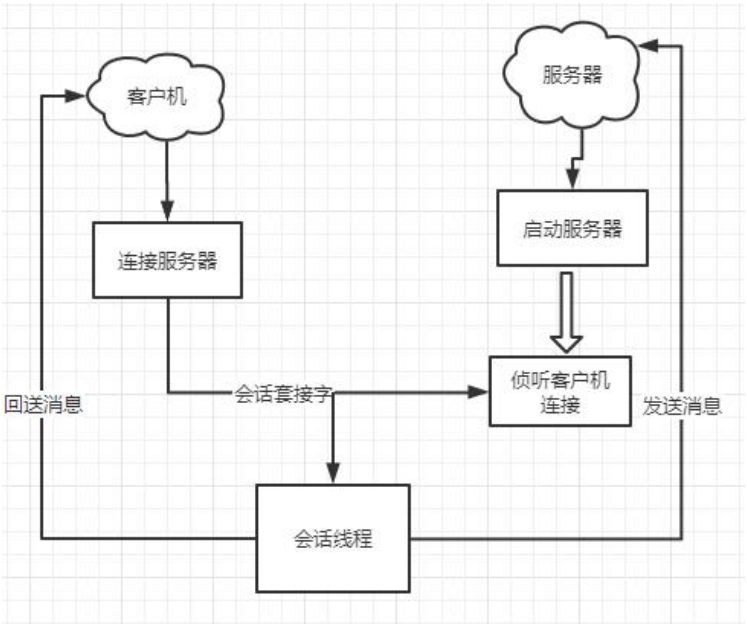
本次实验结构和程序较简单，只有客户机和服务器的 UI 界面以及一个客户机会话线程。

ClientUI:客户机界面，包含消息面板，内有连接服务器和发送消息线程。

ServerUI:服务器界面，启动服务器事件和侦听线程

ClientThread:客户机会话线程，处理和客户机间的消息和通信

程序流程图



部分关键代码

客户机连接服务器

```
// TODO add your handling code here:
try {
    String remoteName = txtRemoteName.getText();
    int remotePort = Integer.parseInt(txtRemotePort.getText());

    SocketAddress remoteAddr = new InetSocketAddress(InetAddress.getByName(remoteName),remotePort);
    clientSocket = new Socket();
    clientSocket.connect(remoteAddr);
    txtArea.append("连接服务器成功,会话开始....\n");
    // 创建绑定到套接字clientSocket上的网络输入流和输出流 true表示追加内容
    out = new PrintWriter(new OutputStreamWriter(clientSocket.getOutputStream(),"UTF-8"),true);
    in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream(),"UTF-8"));
```

获取服务器 ip 地址和端口号，创建会话套接字，构建数据输入输出流。

启动服务器

```
// 禁用按钮，避免重复启动
btnStart.setEnabled(false);
String hostName = txtHostName.getText();// 主机名
int hostPort = Integer.parseInt(txtHostPort.getText()); // 端口

SocketAddress serAddr = new InetSocketAddress(InetAddress.getByName(hostName),hostPort);
listenSocket = new ServerSocket(); // 创建侦听套接字
listenSocket.bind(serAddr); // 绑定到工作地址
txtArea.append("服务器开始等待客户机连接...\n");
```

服务器创建侦听套接字，绑定到服务器相应的工作地址。

```
new Thread(new Runnable() {
    @Override
    public void run() {
        try {
            while(true) {
                toCilentSocket = listenSocket.accept(); // 侦听并接受客户机连接
                clientCounts++; // 客户机数量加1
                txtArea.append(toCilentSocket.getRemoteSocketAddress() + "客户机编号" + clientCounts + "会话开始...\n");
                Thread clientThread = new ClientThread(toCilentSocket,clientCounts);
                clientThread.start(); // 启动任务线程
            }
        }catch(IOException ex) {
            JOptionPane.showMessageDialog(null, ex.getMessage(),"错误提示",JOptionPane.ERROR_MESSAGE);
        }
    }
}).start();
```

在启动服务器的同时，开辟一个新的线程，在这个线程里单独侦听并接收客户机连接，因为 accept()是个阻塞方法，这样就不会导致无客户机连接的情况下导致服务器界面无法操作的情况。

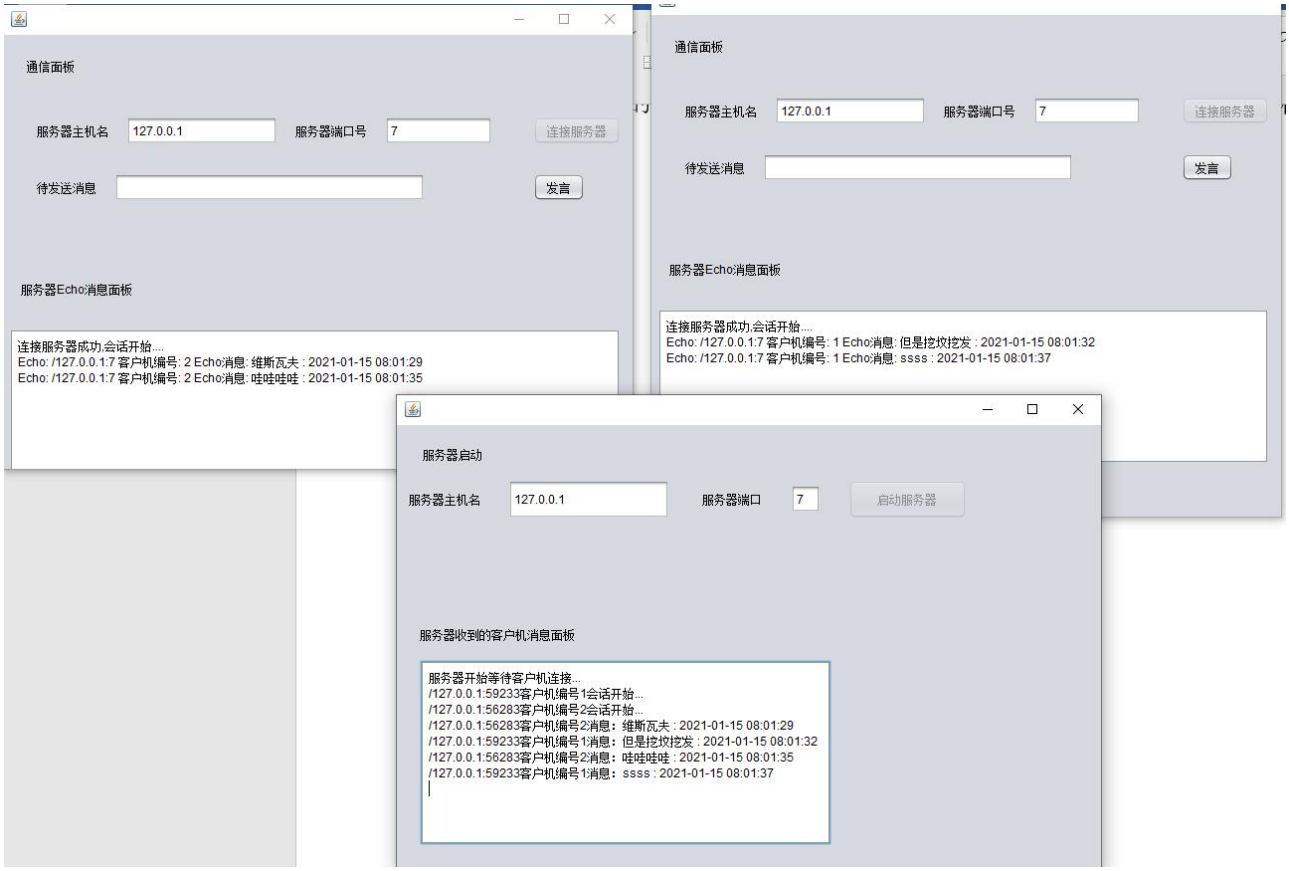
客户机会话线程

```
public ClientThread(Socket toClientSocket,int clientCounts) {
    this.toClientSocket = toClientSocket;
    this.clientCounts = clientCounts;
}

@Override
public void run() {
    try {
        in = new BufferedReader(new InputStreamReader(toClientSocket.getInputStream(),"UTF-8"));
        out = new PrintWriter(new OutputStreamWriter(toClientSocket.getOutputStream(),"UTF-8"),true);
```

在会话线程中传入的两个参数为当前会话套接字和当前客户机连接数量，每一个客户和服务器的会话都分配一个单独的线程，同时都记录和知道了当前客户机连接的数量。

实验运行数据



相关问题

① in.readline()返回空或不返回，分别代表什么？

readLine()是一个阻塞函数，当没有数据读取时，就一直会阻塞在那，而不是返回 null

readLine()只有在数据流发生异常或者另一端被 close()，才会返回 null 值。

readLine 读取一行数据，回车结束，但不会保留回车/换行符。

② Out.flush()应在什么时候使用？

清空缓冲区，把数据全部写入特定的位置(内存)。

③ 如何在二台计算机上进行实验?如何知道对方的 IP ？

通过 ServerSocket 对象然后用 accept()方法接受请求连接的 Socket 对象，在调用 Socket 对象的 getInetAddress()方法获取 InetAddress 对象，再调用 InetAddress 对象的 getHostAddress()方法-→获取 ip 地址。
Socket.getInetAddress().getHostAddress()

④ 程序如何改进？

可以采用 UDP 通信或者 SwingWorker 改写线程。