MÔN : CÔNG NGHỆ JAVA
Bài thực hành 9.1 : Xây dựng chương trình MiniChatClient

I. Mục tiêu :

- Giúp SV làm quen với việc sử dụng môi trường lập trình trực quan NetBeans.
- Giúp SV làm quen với qui trình thiết kế trực quan cửa sổ giao diện chứa nhiều đối tượng giao diện Swing.
- Giúp SV làm quen với việc định nghĩa hàm xử lý sự kiện cho sự kiện xác định của phần tử giao diện xác định.
- Giúp SV làm quen với việc viết code cho module client để kết nối với server, để gởi/nhận request/reply với server.

II. Nội dung :

- Dùng NetBeans để thiết kế cửa sổ giao diện của chương trình MiniChatClient theo slide bài giảng ở chương 9, định nghĩa hàm xử lý sự kiện cho sự kiện của các phần tử giao diện xác định, viết code chức năng cho hàm xử lý, chạy thử phần mềm để kiểm tra kết quả.

III. Chuẩn đầu ra :

- Sinh viên nắm vững việc sử dụng môi trường lập trình trực quan NetBeans để thiết kế cửa sổ giao diện của chương trình, định nghĩa hàm xử lý sự kiện cho sự kiện của phần tử giao diện xác định, viết code chức năng cho hàm xử lý.
- Sinh viên nắm vững việc viết code cho module client để kết nối với server, để gởi/nhận request/reply với server.

IV. Qui trình :

1. Chạy NetBean 7.4, nếu cửa sổ Project có hiển thị các Project cũ hãy đóng chúng lại.
2. Chọn menu File.New Project để máy hiển thị cửa sổ "New Project", chọn mục "Java" trong Listbox Categories, chọn mục "Java Application" trong Listbox Projects rồi click button Next để hiển thị cửa sổ "New Java Application".
3. Xác định thư mục chứa Project ở textbox "Project Location", nhập "MiniChatClient" vào textbox "Project Name", click button Finish để máy tạo thực sự Project. Cửa sổ mã nguồn của class chương trình MiniChatClient hiển thị. Soạn code cho hàm main như sau :

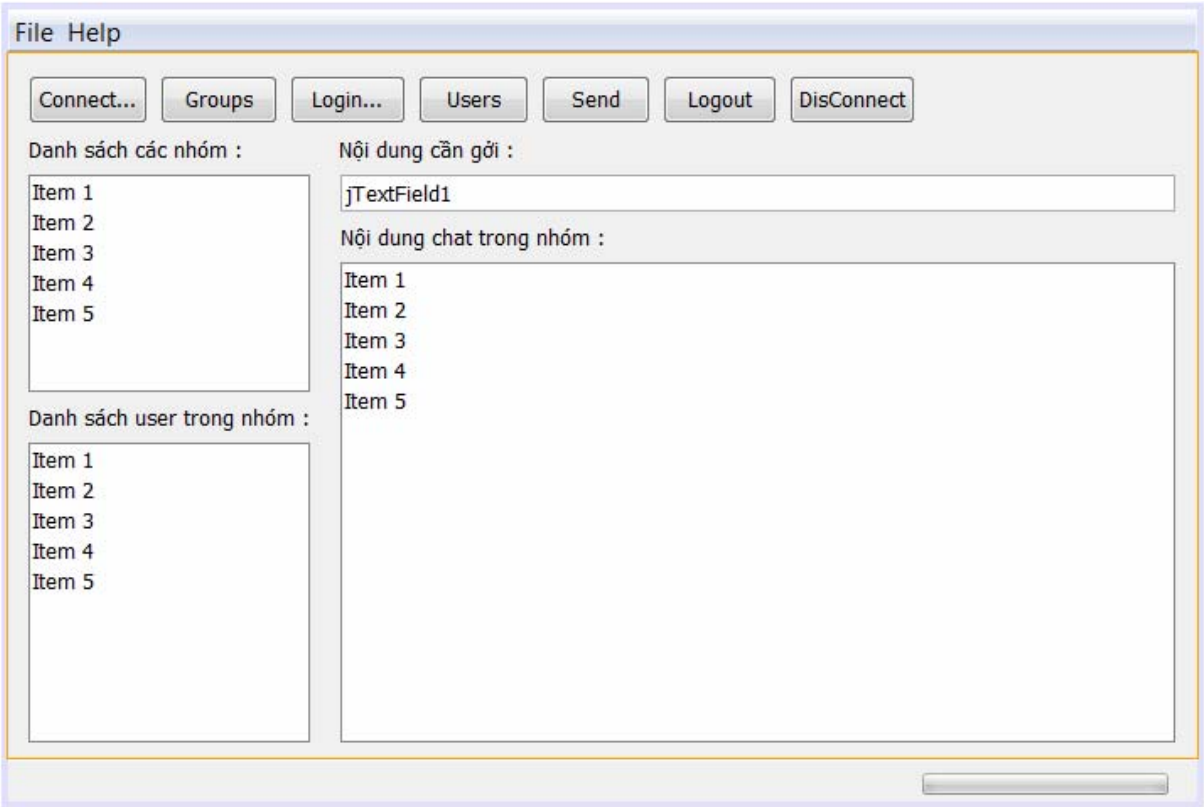
```
public static void main(String[] args) {
    //tạo và hiển thị Form giao diện cho ứng dụng
    MiniChatClientDlg dlg = new MiniChatClientDlg();
    dlg.show();
}
```

4. Trong cửa sổ quản lý Project, ấn kép chuột vào phần tử gốc có tên là MiniChatClient để mở rộng nội dung của nó, bạn sẽ thấy folder "Source Packages", ấn kép chuột vào folder "Source Packages" bạn sẽ thấy folder minichatclient. Ấn phải chuột trên folder minichatclient, chọn chức năng New.JFrame Form để máy hiển thị cửa sổ "New JFrame Form", nhập tên class mới là MiniChatClientDlg, click chuột vào button Finish để máy tạo ra Form tương ứng, cửa sổ thiết kế Form sẽ hiển thị.
5. Chọn icon Button trong cửa sổ Palette (thường ở góc trên phải màn hình), dời chuột về vị trí trên trái cửa sổ thiết kế và vẽ nó, nếu kích thước chưa phù hợp thì hãy hiệu chỉnh lại kích thước theo yêu cầu. Quan sát cây quản lý các đối tượng giao diện ở cửa sổ Navigator (thường nằm ở góc dưới trái màn hình), ta thấy Button mới tạo có tên là jButton1. Ấn phải chuột vào jButton1 để hiển thị menu lệnh kết hợp, chọn mục "Exit Text"

và hiệu chỉnh lại chuỗi hiển thị trên button là "Connect...". Ấn phải chuột vào jButton1 để hiển thị menu lệnh kết hợp, chọn mục "Change Variable Name" và hiệu chỉnh lại tên nhận dạng button là "btnConnect", đây là tên biến để code chương trình truy xuất Button. Duyệt tìm thuộc tính margin trong cửa sổ thuộc tính của Button rồi hiệu chỉnh lại giá trị thành [4,4,4,4].

6. Lặp lại bước 5 nhiều lần để tạo các phần tử giao diện lần lượt như sau :
   - button "Groups" có tên là btnGroups.
   - button "Login" có tên là btnLogin.
   - button "Users" có tên là btnUsers.
   - button "Send" có tên là btnSend.
   - button "Logout" có tên là btnLogout.
   - button "DisConnect" có tên là btnDisConnect.
   - Label "Danh sách các nhóm" có tên mặc định.
   - List có tên là lbGroups.
   - Label "Danh sách các user trong nhóm" có tên mặc định.
   - List có tên là lbUsers.
   - Label "Nội dung cần gởi :" có tên mặc định.
   - TextField có tên là txtMessage.
   - Label "Nội dung chat trong nhóm :" có tên mặc định.
   - List có tên là lbContent.

   Kết quả cửa sổ chương trình có dạng :



7. Chọn button "Connect..." để hiển thị cửa sổ thuộc tính của nó (thường ở góc dưới phải màn hình). Click button Event trong cửa sổ thuộc tính để máy hiển thị danh sách các sự kiện kết hợp với button. Tìm sự kiện actionPerformed, ấn chuột vào mũi tên chỉ xuống trong listbox bên phải sự kiện và chọn hàm btnConnectActionPerformed để máy tạo hàm xử lý sự kiện tương ứng với button. Cửa sổ mã nguồn hiển thị hàm vừa tạo ra với thân rỗng, chúng ta sẽ viết code cho hàm để thực hiện chức năng nối kết đến server chat sau.

Bây giờ click vào button Design nằm ngay trên cửa sổ mã nguồn để hiển thị lại cửa sổ thiết kế.

8. Lặp lại bước 7 nhiều lần để tạo các hàm xử lý sự kiện Click chuột cho các button còn lại.
9. Chọn JTextField để hiển thị cửa sổ thuộc tính của nó. Click button Event trong cửa sổ thuộc tính để máy hiển thị danh sách các sự kiện kết hợp với JTextField. Tìm sự kiện KeyPressed, ấn chuột vào mũi tên chỉ xuống trong listbox bên phải sự kiện và chọn hàm txtMessageKeyPressed để máy tạo hàm xử lý sự kiện tương ứng với JTextField. Cửa sổ mã nguồn hiển thị hàm vừa tạo ra với thân rỗng, chúng ta sẽ viết code cho hàm để thực hiện chức năng gởi thông báo chat đến các thành viên cùng nhóm.
10. Dời chuột về mục minichatclient trong cây Project, ấn kép chuột vào nó để hiển thị menu lệnh, chọn option New.Java Interface để hiển thị cửa sổ "New Java Interface". Nhập tên “MessageListener” vào textbox "Class name' rồi click button Finish để máy tạo thực sự interface mới theo yêu cầu. Cửa sổ soạn mã nguồn của interface được hiển thị, hãy soạn nội dung của interface như sau :

```
package minichatclient;
import java.net.*;

public interface MessageListener {
   //định nghĩa tác vụ được cung cấp bởi interface
   void messageReceived(Socket socket, String s);
}
```

11. Dời chuột về mục minichatclient trong cây Project, ấn kép chuột vào nó để hiển thị menu lệnh, chọn option New.Java Class để hiển thị cửa sổ "New Java Class". Nhập tên “ReceivingThread” vào textbox "Class name' rồi click button Finish để máy tạo thực sự class mới theo yêu cầu. Cửa sổ soạn mã nguồn của class được hiển thị, hãy soạn nội dung của class như sau :

```
package minichatclient;
import java.io.*;
import java.net.*;
import java.util.StringTokenizer;

public class ReceivingThread extends Thread {
   //định nghĩa các thuộc tính cần dùng
   private BufferedReader input;
   private MessageListener messageListener;
   private boolean keepListening = true;
   Socket socket;
   //định nghĩa constructor
   public ReceivingThread(MessageListener listener, Socket clientSocket) {
      //gọi cha
      super("ReceivingThread: " + clientSocket);
      //lưu giữ tham số nhận được từ client gọi
      messageListener = listener;
      socket = clientSocket;
      try {
         //thiết lập timer cho socket
         socket.setSoTimeout(0);
         //tạo đối tượng phục vụ đọc dữ liệu từ xa gởi về
```

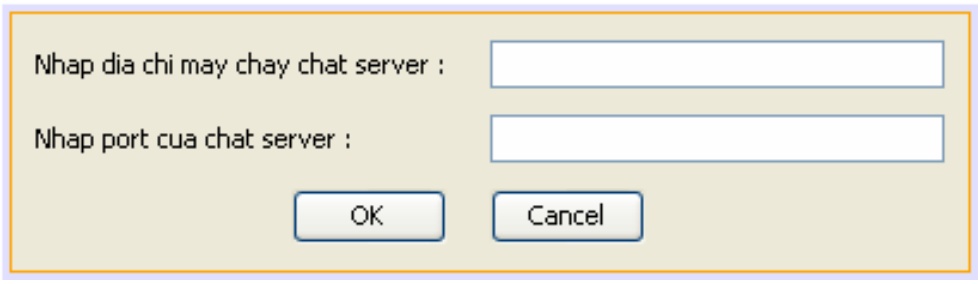
```
            input = new BufferedReader(new InputStreamReader(
                    socket.getInputStream()));
        } //xử lý lỗi ngoại lệ
        catch (IOException ioException) {
            ioException.printStackTrace();
        }
    } //hết constructor

    //thuật giải của thread : chờ nhận thông tin và gởi về MessageListener xử lý
    public void run() {
        String message;
        //chờ nhận thông tin đến khi bị stop
        while (keepListening) {
            try {
              //thử đọc 1 thông báo (=1 dòng văn bản)
              message = input.readLine();
              if (message != null) //nếu có thì gởi cho MessageListener xử lý
               messageListener.messageReceived(socket, message);
            }
            //xử lý lỗi ngoại lệ
            catch (InterruptedIOException interruptedIOException) {
                //tiếp tục lặp chờ
                continue;
            }
            catch (IOException ioException) {
                messageListener.messageReceived(
                        socket, // user name
                        "CLOSE "); // message body
                break;
            }
        } //hết vòng lặp chờ
        try {
            //đóng đối tượng input
            input.close();
        } //xử lý lỗi ngoại lệ
        catch (IOException ioException) {
            ioException.printStackTrace();
        }
    } //hết hàm run

    //hàm thiết lập trạng thái dừng việc chờ
    public void stopListening() {
        keepListening = false;
    }
}//hết hàm class ReceivingThread
```

12. Dời chuột về mục minichatclient trong cây Project, ấn kép chuột vào nó để hiển thị menu lệnh, chọn option New.Others để hiển thị cửa sổ "New File", duyệt tìm và chọn mục "Swing GUI Forms" trong listbox Categories, chọn mục "JDialog Form" trong listbox "File Type", chọn button Next để hiển thị cửa sổ "New JDialog Form". Nhập tên “ConnectDlg”

vào textbox "Class name" rồi click button Finish để máy tạo thực sự Dialog mới theo yêu cầu, form thiết kế trống sẽ hiển thị, bạn hãy thiết kế form theo hình sau :



Form gồm 2 label, 2 JTextField có tên nhận dạng lần lượt là txtAddress và txtPort có nội dung ban đầu là chuỗi rỗng, 2 Button có tên lần lượt là btnOK và btnCancel.

13. Khai báo hàm xử lý sự kiện actionPerformed trên button OK là btnOKActionPerformed, hàm xử lý sự kiện actionPerformed trên button Cancel là btncancelActionPerformed. Viết code cho các hàm xử lý sự kiện vừa tạo ở bước 12 như sau :

```
//định nghĩa hàm xử lý sự kiện btnOKActionPerformed
private void btnOKActionPerformed(java.awt.event.ActionEvent evt) {
  fOk = true;
  intPort = Integer.valueOf(txtPort.getText());
  strAddress = txtAddress.getText();
  this.setVisible(false);
}

//định nghĩa hàm xử lý sự kiện btnCancelActionPerformed
private void btnCancelActionPerformed(java.awt.event.ActionEvent evt) {
  fOk = false;
  this.setVisible(false);
}
```

14. Viết thêm đoạn code sau vào đầu class ConnectDlg để định nghĩa các thuộc tính cần dùng cho Form :

```
//định nghĩa các thuộc tính cần dùng
boolean fOk = false;
public String strAddress;
public int intPort;
```

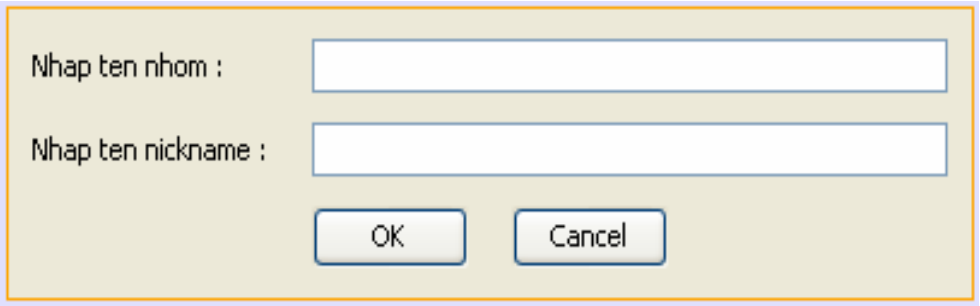
15. Hiệu chỉnh lại hàm constructor cho class ConnectDlg như sau :

```
//hàm constructor cho class ConnectDlg
public ConnectDlg(java.awt.Frame parent, boolean modal, String title) {
  super(parent, modal);
  initComponents();
  this.setTitle(title);
}
```

Duyệt tìm hàm main() của class ConnectDlg, hiệu chỉnh lệnh new ConnectDlg thành :

```
ConnectDlg dialog = new ConnectDlg(new javax.swing.JFrame(), true,"");
```

16. dời chuột về mục minichatclient trong cây Project, ấn kép chuột vào nó để hiển thị menu lệnh, chọn option New.JDialog Form để hiển thị cửa sổ "New JDialog Form". Nhập tên “LoginDlg” vào textbox "Class name' rồi click button Finish để máy tạo thực sự Dialog mới theo yêu cầu, form thiết kế trống sẽ hiển thị, bạn hãy thiết kế form theo hình sau :



Form gồm 2 label, 2 JTextField có tên nhận dạng lần lượt là txtGroup và txtUser có nội dung ban đầu là chuỗi rỗng, 2 Button có tên lần lượt là btnOK và btnCancel.

17. Khai báo hàm xử lý sự kiện actionPerformed trên button OK là btnOKActionPerformed, hàm xử lý sự kiện actionPerformed trên button Cancel là btnCancelActionPerformed. Viết code cho các hàm xử lý sự kiện vừa tạo ở bước 16 như sau :

```
//định nghĩa hàm xử lý sự kiện btnOKActionPerformed
private void btnOKActionPerformed(java.awt.event.ActionEvent evt) {
  fOk = true;
  GroupName = txtGroup.getText();
  UserName = txtUser.getText();
  this.setVisible(false);
}

//định nghĩa hàm xử lý sự kiện btnCancelActionPerformed
private void btnCancelActionPerformed(java.awt.event.ActionEvent evt) {
  fOk = false;
  this.setVisible(false);
}
```

18. Viết thêm đoạn code sau vào đầu class LoginDlg để định nghĩa các thuộc tính cần dùng cho Form :

```
//định nghĩa các thuộc tính cần dùng
   boolean fOk = false;
   String GroupName;
   String UserName;
```

19. Hiệu chỉnh lại hàm constructor cho class LoginDlg như sau :

```
//hàm constructor cho class LoginDlg
public LoginDlg(java.awt.Frame parent, boolean modal, String title, String gname) {
  super(parent, modal);
  initComponents();
  this.setTitle(title);
  txtGroup.setText(gname);
}
```

Duyệt tìm hàm main() của class LoginDlg, hiệu chỉnh lệnh new ConnectDlg thành :

```
LoginDlg dialog = new LoginDlg(new javax.swing.JFrame(), true,"","");
```

20. Dời chuột về mục MiniChatClientDlg.java trong cây Project, ấn kép chuột vào nó để hiển thị cửa sổ soạn mã nguồn cho form giao diện này (nếu cửa sổ soạn mã chưa hiển thị, click chuột vào button Source nằm ngay trên cửa sổ), rồi viết code cho các hàm xử lý sự kiện như sau :

```
//định nghĩa hàm xử lý Click button Connect
private void btnConnectActionPerformed(java.awt.event.ActionEvent evt) {
    //tạo form nhập thông tin về chat server
```

```
    ConnectDlg dlg = new ConnectDlg(new javax.swing.JFrame(), true, "Nhập thông tin về
    server");
    dlg.show();
    if (dlg.fOk) {  //nếu người dùng click OK thì chạy tiếp
        try {
           //tạo socket nối kết chat server
           socket = new Socket(InetAddress.getByName(dlg.strAddress), dlg.intPort);
           //tạo và chạy thread chờ nhận reply từ chat server
           receivingThread = new ReceivingThread(this, socket);
           receivingThread.start();
           //tạo đối tượng phục vụ gởi request đến server
           writer =  new PrintWriter(socket.getOutputStream());
        } //xử lý lỗi ngoại lệ
        catch (Exception e) {
           e.printStackTrace();
        }
        //thiết lập trạng thái đã kết nối
        connected = true;
      }
}

//định nghĩa hàm xử lý Click button Groups
private void btnGroupsActionPerformed(java.awt.event.ActionEvent evt) {
      String mesg = "GLIST ";
      SendMessage(writer, mesg);
      fstate = FSGLIST;
}

//định nghĩa hàm xử lý Click button Login
private void btnLoginActionPerformed(java.awt.event.ActionEvent evt) {
    //tạo form nhập thông tin về user
    LoginDlg dlg;
    if (lbGroups.getSelectedValue().toString() == null) {
        dlg = new LoginDlg(new javax.swing.JFrame(), true,"Nhập thông tin login", "");
    } else {
        dlg = new LoginDlg(new javax.swing.JFrame(), true,"Nhập thông tin login",
    lbGroups.getSelectedValue().toString());
    }
    dlg.show();
    if (dlg.fOk) { //nếu người dùng click OK thì chạy tiếp
        String request;
        request = "LOGIN " + dlg.GroupName + "," + dlg.UserName;
        SendMessage(writer, request);
        fstate = FSLOGIN;
      }
}

//định nghĩa hàm xử lý Click button Users
private void btnUsersActionPerformed(java.awt.event.ActionEvent evt) {
      String mesg = "ULIST ";
```

```
        SendMessage(writer, mesg);
        fstate = FSULIST;
}

//định nghĩa hàm xử lý Click button Send
private void btnSendActionPerformed(java.awt.event.ActionEvent evt) {
            SendToGroup();
}

//định nghĩa hàm xử lý Click button Logout
private void btnLogoutActionPerformed(java.awt.event.ActionEvent evt) {
       String mesg;
        mesg = "LOGOU ";
        SendMessage(writer, mesg);
        fstate = FSLOGOUT;
}

//định nghĩa hàm xử lý Click button Disconnect
private void btnDisConnectActionPerformed(java.awt.event.ActionEvent evt) {
       try {
            SendMessage(writer, "LOGOU ");
            fstate = FSLOGOUT;
        } catch (Exception e) {
            e.printStackTrace();
        }
}

//định nghĩa hàm xử lý nhập ký tự trên JTextField
private void txtMessageKeyPressed(java.awt.event.KeyEvent evt) {
     //nếu user gỏ Enter thì gởi chuỗi vừa nhập đi
     if (evt.getKeyCode()==10)   SendToGroup();
}
```

21. Viết tiếp các hàm dịch vụ như sau :

```
//hàm gởi 1 request đến server
    private void SendMessage(PrintWriter writer, String mesg) {
        //hàm gởi 1 request đến server rồiflush PrintWriter
        try {
            writer.println(mesg);
            writer.flush();
        } //xử lý lỗi ngoại lệ
        catch (Exception e) {
            e.printStackTrace();
        }
    }
//hàm gởi 1 message đến các thành viên của nhóm (thông qua server)
    private void SendToGroup() {
        String mesg = txtMessage.getText();
        if (mesg != "") {
            SendMessage(writer, "SEND  " + mesg);
        }
```

```
        fstate = FSMESG;
    }
//hàm xử lý reply của server
    public void messageReceived(Socket sock, String mesg) {
        int status;
        if (mesg.compareTo("CLOSE ") == 0) {
            return;
        }
        switch (fstate) { //xử lý reply theo trạng thái lệnh request hiện hành
            case FSLOGIN:
                Do_login(mesg);
                break;
            case FSLOGOUT: // logout
                Do_logout(mesg);
                break;
            case FSMESG: // group list
                Do_receive(mesg);
                break;
            case FSGLIST: // user list
                Do_glist(mesg);
                break;
            case FSULIST: // broadcast message
                Do_ulist(mesg);
        }
        fstate = FSMESG;
    }

//hàm xử lý reply của request Login
    private void Do_login(String mesg) {
    }

//hàm xử lý reply của request Logout
    private void Do_logout(String mesg) {
        if (mesg.charAt(0) == '0') {
            return;
        }
        DefaultListModel lmUsers = (DefaultListModel) lbUsers.getModel();
        lmUsers.clear();
        DefaultListModel lmContent = (DefaultListModel) lbContent.getModel();
        lmContent.clear();
    }

//hàm xử lý reply của request GLIST
    private void Do_glist(String mesg) {
        int i;
        DefaultListModel lmGroups = (DefaultListModel) lbGroups.getModel();
        lmGroups.clear();
        if (mesg.charAt(0) == '0') {
            return;
        }
```

```
        mesg = mesg.substring(2);
        // tokenize message to retrieve user name and message body
        java.util.StringTokenizer tokenizer = new java.util.StringTokenizer(mesg, ",");
        int size = tokenizer.countTokens();
        for (i = 0; i < size; i++) {
            lmGroups.addElement(tokenizer.nextToken());
        }
    }
//hàm xử lý reply của request ULIST
    private void Do_ulist(String mesg) {
        String gname;
        int i;
        DefaultListModel lmUsers = (DefaultListModel) lbUsers.getModel();
        lmUsers.clear();
        if (mesg.charAt(0) == '0') {
            return;
        }
        if (mesg.length() < 3) {
            return;
        }
        mesg = mesg.substring(2);
        // tokenize message to retrieve user name and message body
        java.util.StringTokenizer tokenizer = new java.util.StringTokenizer(mesg, ",");
        int size = tokenizer.countTokens();
        for (i = 0; i < size; i++) {
            lmUsers.addElement(tokenizer.nextToken());
        }
    }

//hàm xử lý reply của request MESG
    private void Do_receive(String mesg) {
        DefaultListModel lmContent = (DefaultListModel) lbContent.getModel();
        // append new message
        lmContent.addElement(mesg);
    }
```

22. Dời cursor về đầu file, thêm các lệnh import sau vào (ngay sau các lệnh import đã có) :

```
import javax.swing.*;
import java.net.*;
import java.io.*;
```

23. Hiệu chỉnh lại lệnh định nghĩa class như sau :

```
public class MiniChatClientDlg extends javax.swing.JFrame implements
MessageListener {
```

24. Thêm các lệnh định nghĩa các thuộc tính cần dùng như sau :

```
    private Socket socket;   //socket giao tiếp với server
    private boolean connected = false;  //cờ trạng thái làm việc của client
    private ReceivingThread receivingThread; //handle đến thread con chờ reply
    //các hằng miêu tả request hiện hành
    private static final int FSGLIST = 0;
    private static final int FSLOGIN = 1;
```

```
private static final int FSLOGOUT = 2;
private static final int FSULIST = 3;
private static final int FSMESG = 4;
private int fstate = FSMESG;
PrintWriter writer;
```

25. Dời cursor về cuối hàm constructor của MiniChatClientDlg, thêm các lệnh thiết lập giá trị đầu của các đối tượng giao diện :

```
//xóa nội dung các List và JTextField
txtMessage.setText("");
DefaultListModel model = new DefaultListModel();
lbGroups.setModel((ListModel) model);
model = new DefaultListModel();
lbUsers.setModel((ListModel) model);
model = new DefaultListModel();
lbContent.setModel((ListModel) model);
```

26. Chọn menu Run.Run Project để dịch và chạy thử chương trình. Nếu có lỗi từ vựng và cú pháp thì sữa, nếu có lỗi run-time thì debug (thông qua các chức năng trong menu Debug) để xác định lỗi rồi sữa lỗi.

27. Nếu chương trình hết lỗi, thực hiện bài 9.1 để xây dựng module MiniChatServer để phục vụ chạy module client này.