MÔN : CÔNG NGHỆ JAVA
Bài thực hành 10.1 : Xây dựng chương trình MiniChatServer

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 server lắng nghe yêu cầu kết nối từ các client, để nhận/gởi request/reply với từng client.

II. Nội dung :

- Dùng NetBeans để thiết kế cửa sổ giao diện của chương trình MiniChatServer theo slide bài giảng ở chương 10, 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 server lắng nghe yêu cầu kết nối từ các client, để nhận/gởi request/reply với từng client.

IV. Qui trình :

IV.1 Tạo database chứa danh sách nhóm tán gẫu mà server cần quản lý :

1. Chạy Access, tạo 1 table (với tên là GroupList) gồm nhiều record, mỗi record miêu tả thông tin về 1 nhóm, để đơn giản ta chỉ cần 1 field cho record có tên là groupname.
2. Thêm lần lượt từng nhóm (nhập tên nhóm) vào table, lưu kết quả vào file *.mdb.
3. Chạy tiện ích tạo DSN quản lý database (nếu dùng driver 64bit thì chạy tiện ích c:\windows\system32\odbcad32.exe, nếu dùng driverf 32bit thì chạy tiện ích c:\windows\sysWOW64\odbcad32.exe), tạo 1 DSN thuộc loại System DSN, dùng driver Microsoft database Access (*.mdb), đặt tên cho DSN là Grouplist, duyệt tìm và chọn file database vừa tạo ra ở bước 2.

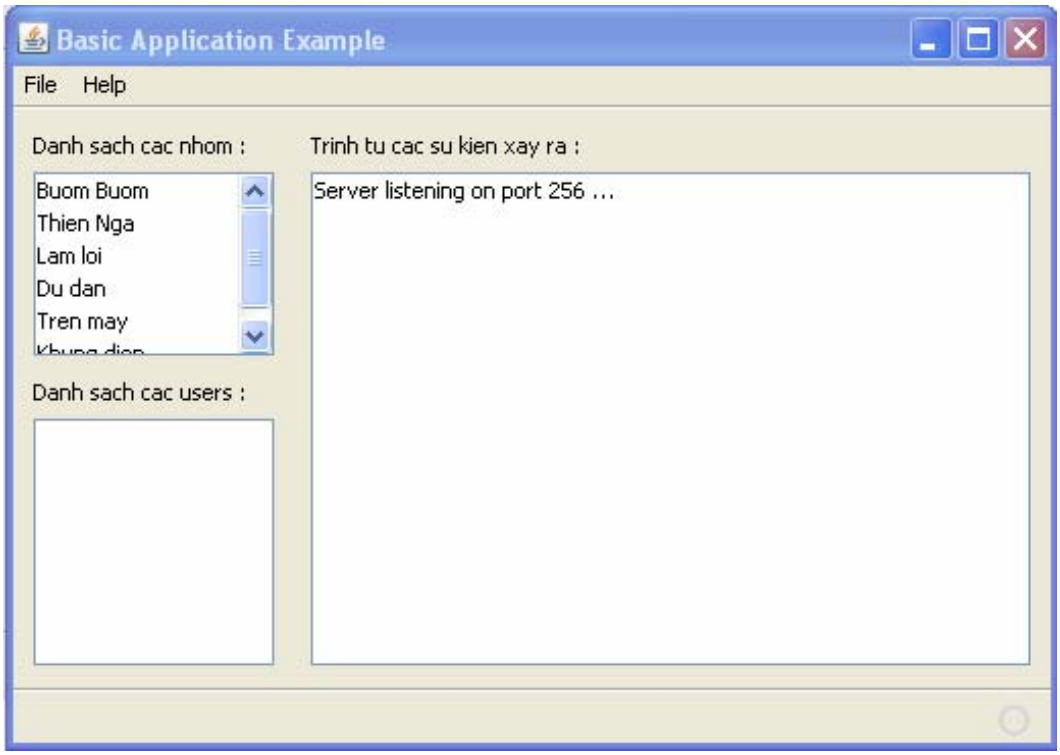
IV.2 Xây dựng module server bằng NetBeans :

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 "MiniChatServer" vào textbox "Project Name", click button Finish để máy tạo thực sự Project. Cửa sổ mã nguồn của class chương trình MiniChatServer 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
    MiniChatServerDlg dlg = new MiniChatServerDlg();
    dlg.show();
```

```
}
```

4. Trong cửa sổ quản lý Project, ấn kép chuột vào phần tử gốc có tên là MiniChatServer để mở rộng nội dung của nó, bạn sẽ thấy folder "Source Packages", ấn kép chuột vào folder "Source Packages" bạn sẽ thấy folder minichatserver. Ấn phải chuột trên folder minichatserver, chọn chức năng New.JFrame Form để máy hiển thị cửa sổ "New JFrame Form", nhập tên class mới là MiniChatServerDlg, click chuột vào button Finish để máy tạo ra Form tương ứng, cửa sổ thiết kế Form sẽ hiển thị. Thiết kế form ứng dụng theo hình sau :



Luu ý các phần tử giao diện lần lượt như sau :

- Label có nội dung "Danh sách các nhóm :".
- List có tên là lbGroups.
- Label có nội dung "Danh sách các user trong nhóm :".
- List có tên là lbUsers.
- Label có nội dung "Trình tự các sự kiện xảy ra :".
- List có tên là lbContent.

5. Chọn đối tượng lbContent để hiển thị cửa sổ Properties, click button Code, duyệt tìm và hiệu chỉnh thuộc tính "Variable Modifiers" thành public.

6. Chọn List lbGroups 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 List. Tìm sự kiện mouseClicked, ấn chuột vào mũi tên chỉ xuống trong listbox bên phải sự kiện và chọn hàm lbGroupsMouseClicked để máy tạo hàm xử lý sự kiện tương ứng. 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 hiển thị danh sách user trong nhóm này.

7. dời chuột về mục minichatserver 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 minichatserver;
```

```
import java.net.*;

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

8. dời chuột về mục minichatserver 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 “T_UserRec” 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 minichatserver;
import java.net.*;
//T_UserRec đặc tả record thông tin về 1 user
public class T_UserRec {
   String name;
   Socket sock;
   T_UserRec next;
}
```

9. dời chuột về mục minichatserver 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 “T_GroupList” 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 minichatserver;
//T_GroupList đặc tả record thông tin về 1 nhóm tán gẫu
public class T_GroupList {
  String name;
  T_UserRec userlist;
}
```

10. dời chuột về mục minichatserver 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 minichatserver;
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
```

11. dời chuột về mục minichatserver 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 “ServerAcceptThread” 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 minichatserver;
import java.net.*;
import javax.swing.*;
public class ServerAcceptThread extends Thread {
    //định nghĩa các thuộc tính cần dùng
    ServerSocket serverSocket;
    MiniChatServerDlg serverChat;
    //định nghĩa constructor
    public ServerAcceptThread(MiniChatServerDlg server, ServerSocket sock) {
        serverSocket = sock;
        serverChat = server;
    }

    //thuật giải của thread : chờ nhận yêu cầu nối kết từ client
    public void run() {
        T_UserRec puser;
        try {
            while (true) {
                //chờ 1 yêu cầu
                Socket clientSocket = serverSocket.accept();
                //tạo record chứa các thông tin về yêu cầu
                puser = new T_UserRec();
                puser.sock = clientSocket;
                puser.next = serverChat.m_sock_no_user;
                serverChat.m_sock_no_user = puser;
                //tạo thread chờ nhận request từ client tương ứng
                new ReceivingThread(serverChat, clientSocket).start();
                //hiển thị thông tin về client tương ứng
                DefaultListModel lmContent =
                        (DefaultListModel) serverChat.lbContent.getModel();
                lmContent.addElement("Connection received from: "
                        + clientSocket.getInetAddress());
                serverChat.SendMessage(clientSocket, "Request accepted");
            } // end while
        } //xử lý lỗi ngoại lệ
        catch (Exception e) {
            e.printStackTrace();
        }
    }
}//hết hàm class ServerAcceptThread
```

12. dời chuột về mục MiniChatServerDlg.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 option trong listbox lbGroups
private void lbGroupsMouseClicked(java.awt.event.MouseEvent evt) {
    //xác định tên nhóm được chọn trong listbox lbGroups
    String gname = lbGroups.getSelectedValue().toString();
    int i,j;
    if (gname == null) return;
    //tìm chỉ số nhóm cần tìm trong danh sách nhóm
    for (i = 0; i < m_groupcnt; i++)
    if (gname.compareTo(m_grouplist[i].name) == 0) break;
    if (i >= m_groupcnt) return;
    T_UserRec ulist = m_grouplist[i].userlist;
    DefaultListModel lmUsers = (DefaultListModel)lbUsers.getModel();
    lmUsers.clear();
    while (ulist != null) {
       lmUsers.addElement(ulist.name);
       ulist = ulist.next;
    }
   }
```

13. Viết tiếp các hàm dịch vụ như sau :

```
//hàm đọc danh sách nhóm từ database và hiển thị lên listbox
private void ReadDisplayGroups() {
       String conStr = "jdbc:odbc:GroupList";
       Connection con;
       String newSQL = "Select * from GroupList";
       String[] data = {"dummy"};
       DefaultListModel lmGroups = (DefaultListModel) lbGroups.getModel();
       try {
          Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
          //1. Tao connection mieu ta database can truy xuat
          con = DriverManager.getConnection(conStr, "", "");
          //2. Tao 1 doi tuong Statement lien ket den connection
          java.sql.Statement stmt = con.createStatement();
          //4. Tao doi tuong recordset chua ket qua cua lenh SQL
          ResultSet rs = stmt.executeQuery(newSQL);
          //5. Duyet recordset de xu ly cac record cua no
          int i = 0;
          lmGroups.clear();
          if (rs != null) {
             while (rs.next()) {
                m_grouplist[i] = new T_GroupList();
                m_grouplist[i].name = rs.getString("groupname");
                lmGroups.addElement(m_grouplist[i].name);
                i++;
             }
          }
          m_groupcnt = i;
```

```
            //6. Dong cac dong tuong da tao ra
            rs.close();
            stmt.close();
            con.close();
        } catch (Exception e) {
            System.out.println("Error : " + e);
        }
    }
//hàm xử lý request của client gởi tới từ socket
public void messageReceived(Socket sock, String mesg) {
        int status;
        String opcode = mesg.substring(0, 5);
        if (opcode.compareTo("LOGIN") == 0) { // login
            Do_login(sock, mesg);
        } else if (opcode.compareTo("LOGOU") == 0) { // logout
            Do_logout(sock);
        } else if (opcode.compareTo("GLIST") == 0) { // group list
            Do_glist(sock);
        } else if (opcode.compareTo("ULIST") == 0) { // user list
            Do_ulist(sock);
        } else if (opcode.compareTo("CLOSE") == 0) { // user list
            Do_CloseSock(sock);
        } else { // broadcast message
            Do_MulticastMesg(sock, mesg.substring(5));
        }
    }
//hàm xử lý request GLIST
    private void Do_glist(Socket sock) {
        String mesg, mesg1;
        int i;
        mesg = "1 " + m_grouplist[0].name;
        for (i = 1; i < m_groupcnt; i++) {
            mesg = mesg + "," + m_grouplist[i].name;
        }
        SendMessage(sock, mesg);
    }

//hàm tìm nhóm của client giao tiếp thông qua socket sock
    private int Findgroup(Socket sock) {
        int i;
        T_UserRec pu;
        for (i = 0; i < m_groupcnt; i++) {
            pu = m_grouplist[i].userlist;
            while (pu != null) {
                if (pu.sock == sock) {
                    uname = pu.name;
                    return i;
                }
                pu = pu.next;
            }
```

```
        }
        return -1;
    }
//hàm xử lý request ULIST
private void Do_ulist(Socket sock) {
        String mesg;
        int i = Findgroup(sock);
        if (i < 0 || m_grouplist[i].userlist == null) {
            mesg = "0 ";
        } else {
            T_UserRec pu = m_grouplist[i].userlist;
            mesg = "1 ";
            while (pu != null) {
                mesg = mesg + pu.name;
                pu = pu.next;
                if (pu != null) {
                    mesg = mesg + ",";
                }
            }
        }
        SendMessage(sock, mesg);
    }

//hàm xử lý request MESG
private void Do_MulticastMesg(Socket sock, String mesg) {
        int i = Findgroup(sock);
        if (i < 0) {
            return;
        }
        mesg = uname + ":" + mesg;
        T_UserRec pu = m_grouplist[i].userlist;
        while (pu != null) {
            SendMessage(pu.sock, mesg);
            pu = pu.next;
        }
    }
//hàm đóng socket
    private void Do_CloseSock(Socket sock) {
        int i = Findgroup(sock);
        T_UserRec pu, pup = null;
        if (i >= 0) {
            pu = m_grouplist[i].userlist;
            while (pu != null && pu.sock != sock) {
                pup = pu;
                pu = pu.next;
            }
            if (pu == m_grouplist[i].userlist) {
                m_grouplist[i].userlist = pu.next;
            } else {
                pup.next = pu.next;
```

```
            }
        } else {
            pu = m_sock_no_user;
            while (pu != null && pu.sock != sock) {
                pup = pu;
                pu = pu.next;
            }
            if (pu == m_sock_no_user) {
                m_sock_no_user = pu.next;
            } else {
                pup.next = pu.next;
            }
        }
        try {
            sock.close();
        }
 // handle exception connecting to server
        catch (IOException ioException) {
            ioException.printStackTrace();
        }
    }

//hàm xử lý request LOGIN
    private void Do_login(Socket sock, String mesg) {
        T_UserRec pup, pu;
        String gname, uname;
        int i;
        mesg = mesg.substring(6);
        // tokenize message to retrieve user name and message body
        StringTokenizer tokenizer = new StringTokenizer(mesg, ",");
        // ignore messages that do not contain a user
        // name and message body
        if (tokenizer.countTokens() == 2) {
            gname = tokenizer.nextToken();
            uname = tokenizer.nextToken();
            // tim group tuong ung
            for (i = 0; i < m_groupcnt; i++) {
                if (gname.compareTo(m_grouplist[i].name) == 0) {
                    break;
                }
            }
            System.out.println(gname + " " + uname + "  " + i);
            if (i < m_groupcnt) {
                // tim sokcet trong danh sach chua co user
                pup = pu = m_sock_no_user;
                while (pu != null && pu.sock != sock) {
                    pup = pu;
                    pu = pu.next;
                }
                if (pu != null) {
```

```
                if (pu == m_sock_no_user) {
                    m_sock_no_user = pu.next;
                } else {
                    pup.next = pu.next;
                }
                pu.next = m_grouplist[i].userlist;
                pu.name = uname;
                m_grouplist[i].userlist = pu;
                uname = "1 ";
                SendMessage(sock, uname);
                return;
            }
        }
    }
    errordisp:
    uname = "0 ";
    SendMessage(sock, uname);
  }
//hàm xử lý request LOGOU
  private void Do_logout(Socket sock) {
    int i = Findgroup(sock);
    T_UserRec pu, pup = null;
    if (i >= 0) {
      pu = m_grouplist[i].userlist;
      while (pu != null && pu.sock != sock) {
        pup = pu;
        pu = pu.next;
      }
      if (pu == m_grouplist[i].userlist) {
        m_grouplist[i].userlist = pu.next;
      } else {
        pup.next = pu.next;
      }
      pu.next = m_sock_no_user;
      m_sock_no_user = pu;
      SendMessage(sock, "1 ");
    } else {
      SendMessage(sock, "0 ");
    }
  }

//hàm xử lý chuỗi reply
  public void SendMessage(Socket sock, String mesg) {
    // send message and flush PrintWriter
    try {
      PrintWriter writer = new PrintWriter(sock.getOutputStream());
      writer.println(mesg);
      writer.flush();
    } // handle exception sending message
    catch (IOException ioException) {
```

```
            ioException.printStackTrace();
        }
    }
```

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

```
import javax.swing.*;
import java.util.StringTokenizer;
import java.net.*;
import java.io.*;
import java.sql.*;
```

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

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

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

```
    int m_groupcnt;   //số nhóm
    T_UserRec m_sock_no_user = null;  //danh sách user chưa login
    T_GroupList m_grouplist[] = new T_GroupList[10];   //danh sách nhóm
    String uname;  //tên user
    ServerSocket serverSocket;     //socket server dùng để accept
    final int SERVER_PORT = 256;  //port lắng nghe của server
```

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

```
    //xóa nội dung các List
    DefaultListModel model = new DefaultListModel();
    lbGroups.setModel((ListModel) model);
    model = new DefaultListModel();
    lbUsers.setModel((ListModel) model);
    model = new DefaultListModel();
    lbContent.setModel((ListModel) model);
    //đọc danh sách nhóm từ database và hiển thị
    ReadDisplayGroups();
    //tạo socket để accept cho server
    try {
            serverSocket = new ServerSocket(SERVER_PORT, 100);
            DefaultListModel lmContent = (DefaultListModel) lbContent.getModel();
            lmContent.addElement("Server listening on port " + SERVER_PORT + " ...");
            //tạo thread chuyên chờ accept cho server
            new ServerAcceptThread(this, serverSocket).start();
    }
    //xử lý lỗi ngoại lệ
    catch (IOException ioException) {
            ioException.printStackTrace();
    }
```

18. 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.

19. Nếu chương trình hết lỗi, cửa sổ chương trình sẽ hiển thị, danh sách các nhóm sẽ được hiện thị trong Listbox lbGroups. Hiện server đang chờ yêu cầu nối kết từ các client.

20. Dời cursor về cửa sổ NetBeans, chọn menu File.Open Project để hiển thị cửa sổ "Open Project". Duyệt tìm thư mục chức Project chương trình client đã viết trong bài 9.1 để mở lại Project này.
21. duyệt tìm và ấn phải chuột trên mục MiniChatClient.java trong cây Project để hiển thị menu lệnh, chọn option “Run File” để chạy module client. Cửa sổ chương trình client sẽ hiển thị. Lặp lại bước 21 nhiều lần để chạy nhiều module client đồng thời. Trên mỗi client, hãy thực hiện lần lượt các chức năng Connect, Groups, Login, Users, nhập chuỗi rồi Send... để kiểm tra việc chat giữa các user trong cùng nhóm.