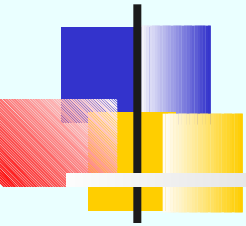
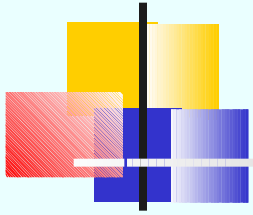


Lập trình mạng

Lập trình Socket với UDP

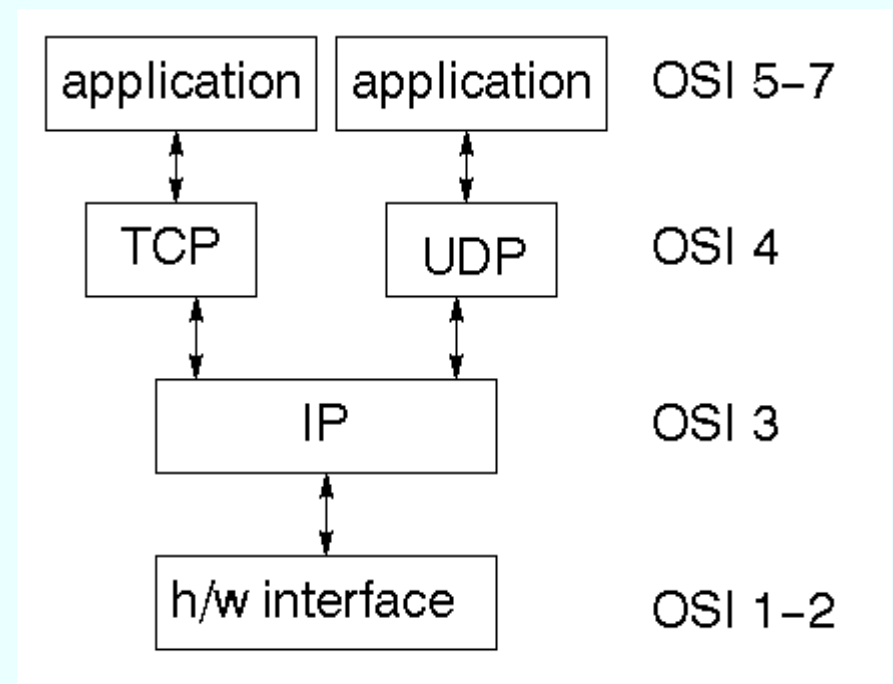
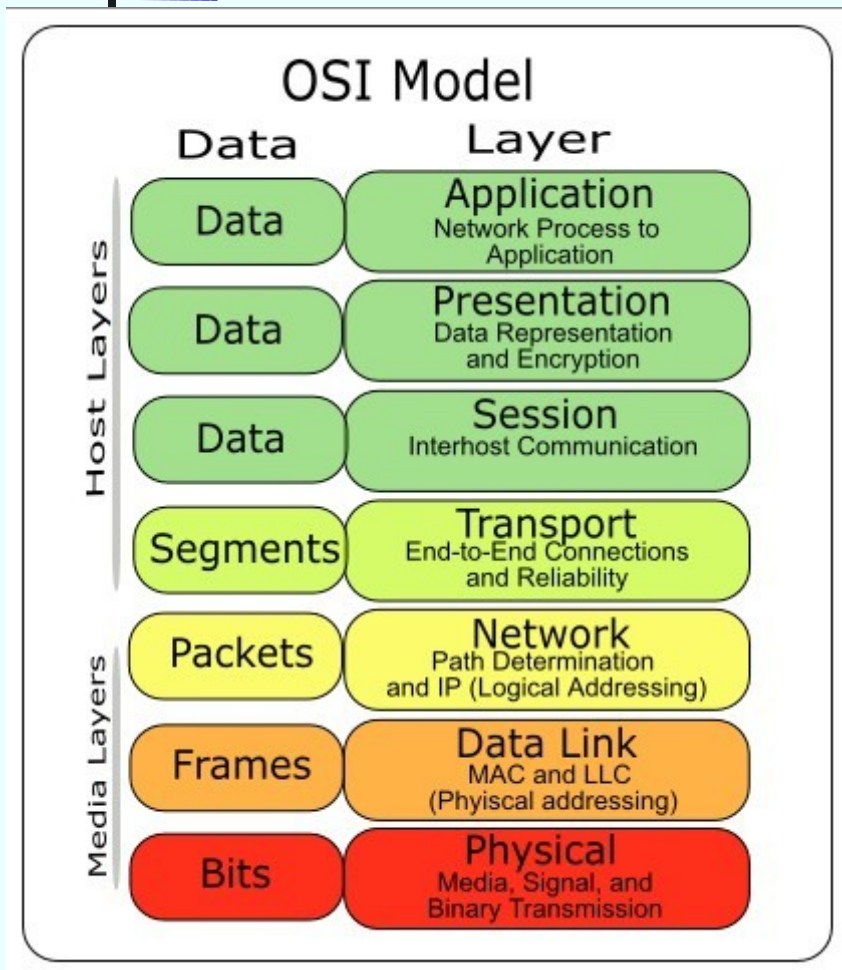




Nội dung

- Giao thức UDP
- Cài đặt phía server
- Cài đặt phía client
- Ví dụ: đảo ngược chuỗi
- Bài tập

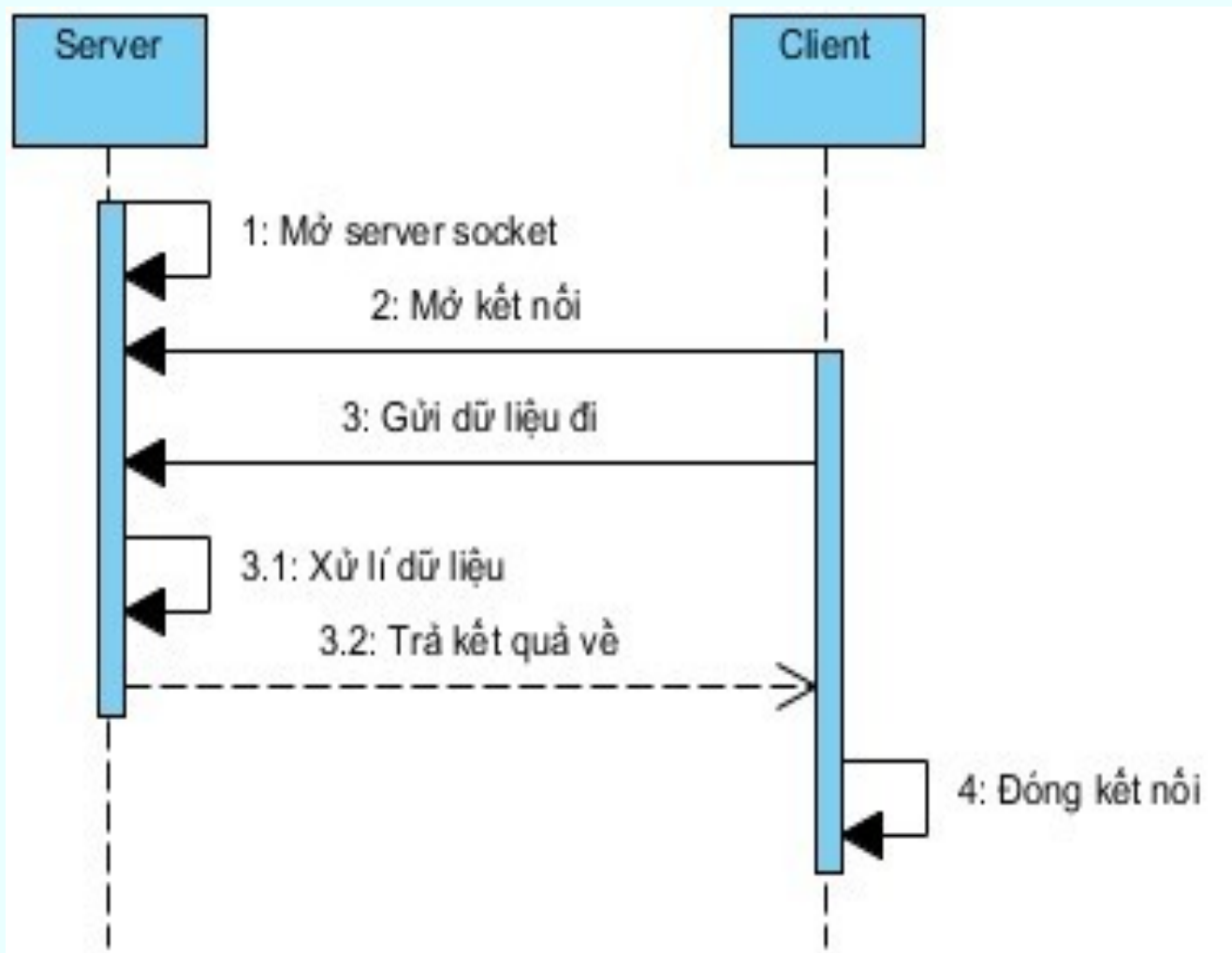
UDP trong mô hình ISO



[image source: <http://1.bp.blogspot.com>]

[image source: <http://jan.newmarch.name>]

Giao thức UDP

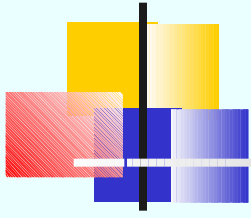


Thiết Kế Giải Thuật Client/Server

- **Giải thuật cho chương trình server iterative, connectionless:**
 - Tạo Socket và đăng ký với hệ thống
 - Lặp công việc đọc dữ liệu từ client gửi đến, xử lý và gọi trả kết quả cho client theo đúng giao thức lớp ứng dụng đã thiết kế

Thiết Kế Giải Thuật Client/Server

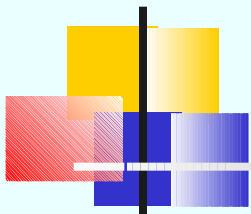
- **Giải thuật cho chương trình server concurrent, connectionless:**
 - Tạo Socket và đăng ký với hệ thống
 - Lặp công việc đọc dữ liệu từ client gửi đến, đối với một dữ liệu nhận, tạo mới một process để xử lý. Tiếp tục nhận dữ liệu mới từ client.
 - Công việc của process mới:
 - Nhận thông tin của process cha chuyển đến, lấy thông tin socket
 - Xử lý và gửi thông tin về cho client theo giao thức lớp ứng dụng đã thiết kế
 - Kết thúc



Server (1)

Bước 1: Mở một server socket tại một cổng có số hiệu xác định

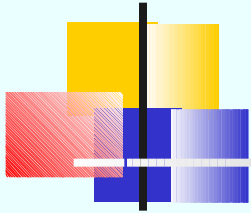
```
try {  
    DatagramSocket myServer = new DatagramSocket(port);  
} catch (SocketException e) {  
    System.out.println(e);  
}
```



Server (2)

Bước 2: Tạo một đối tượng packet từ DatagramPacket để nhận dữ liệu từ phía client để xử lý

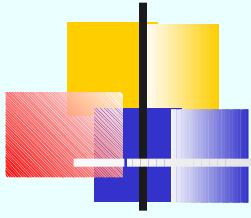
```
try {  
    // Nhan du lieu  
    byte[] receiveData = new byte[1024];  
    DatagramPacket receivePacket = new  
        DatagramPacket(receiveData,  
receiveData.length);  
    myServer.receive(receivePacket);  
    input = new String( receivePacket.getData());  
  
    // Xu li du lieu  
    ...  
} catch (SocketException e) {  
    System.out.println(e);  
} catch (IOException e) {  
    System.out.println(e);  
}
```

Server (3)

Bước 3: Đóng gói thông tin vào gói tin DatagramPacket để gửi trả về cho client tương ứng

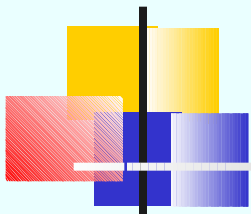
```
try {  
    // Đóng gói thông tin dữ liệu cần trả lại  
    InetAddress IPAddress = receivePacket.getAddress();  
    int port = receivePacket.getPort();  
    byte[] sendData = (dữ liệu đã xử lý).getBytes();  
    DatagramPacket sendPacket = new  
    DatagramPacket(sendData, sendData.length, IPAddress,  
    port);  
  
    // Gửi dữ liệu về client  
    myServer.send(sendPacket);  
} catch (SocketException e) {  
    System.out.println(e);  
} catch (IOException e) {  
    System.out.println(e);  
}
```



Client (1)

Bước 1: Mở một client socket đến server có tên xác định, tại một cổng có số hiệu xác định

```
try {  
    mySocket = new DatagramSocket(clientPort) ;  
} catch (SocketException e) {  
    System.err.println(e) ;  
}
```

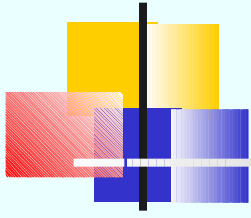


Client (2)

Bước 2: đóng gói thông tin vào gói tin DatagramPacket để gửi đi

```
byte[] sendData = new byte[1024]; // bộ đệm gửi dữ liệu
try {
    InetAddress IPAddress =
        InetAddress.getByName("localhost");

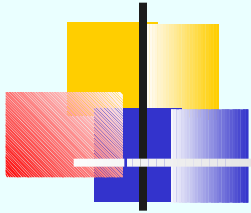
    sendData = (dữ liệu gửi).getBytes();
    DatagramPacket sendPacket = new
        DatagramPacket(sendData,
            sendData.length, IPAddress, số
cổng);
} catch (SocketException e) {
    System.err.println(e);
} catch (IOException e) {
    System.err.println(e);
}
```



Client (3)

Bước 3: Gửi dữ liệu đến server

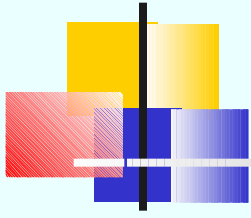
```
try {  
    mySocket.send(sendPacket) ;  
} catch (SocketException e) {  
    System.err.println(e) ;  
} catch (IOException e) {  
    System.err.println(e) ;  
}
```



Client (4)

Bước 4: Nhận dữ liệu đã qua xử lí từ server về

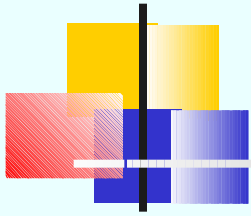
```
byte[] receiveData = new byte[1024]; // bo dem nhan du lieu
try {
    DatagramPacket receivePacket = new
    DatagramPacket(receiveData, receiveData.length);
    mySocket.receive(receivePacket);
    dữ liệu nhận được = receivePacket.getData();
} catch (SocketException e) {
    System.err.println(e);
} catch (IOException e) {
    System.err.println(e);
}
```



Client (5)

Bước 5: Đóng các kết nối tới server

```
        try {  
            mySocket.close();  
        } catch (Exception e)  
        {  
            System.err.println(e);  
        }
```



Ví dụ: đảo chuỗi (1)

```
import java.lang.String;

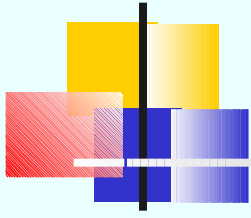
public class ReverseString {
    private String _string;

    // khởi tạo không tham số
    public ReverseString() {
        super();
    }

    // khởi tạo có tham số
    public ReverseString(String
        _string) {
        super();
        this._string = _string;
    }

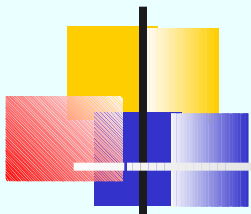
    public String get_string() {
        return _string;
    }

    public void set_string(String
        _string) {
        this._string = _string;
    }
}
```



Ví dụ: đảo chuỗi (2)

```
//phuong thuc dao nguoc chuoì kí tu của lớp này
public void reverse(){
    String tmp ="";
    for(int i=_string.length() - 1; i >=0 ;i--)
        tmp += _string.substring(i, i+1);
    this._string = tmp;
}
}
```

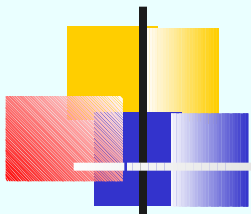



Ví dụ: đảo chuỗi – server (1)

```
import java.net.DatagramSocket;
import java.net.DatagramPacket;
import java.net.InetAddress;
import java.net.SocketException;
import java.io.IOException;

public class UDPServer {
    DatagramSocket myServer = null;
    String input;
    int port = 9900;

    // Mở một server socket
    public void openServer() {
        try {
            myServer = new
DatagramSocket(port);
        } catch (SocketException e) {
            System.out.println(e);
        }
    }
}
```

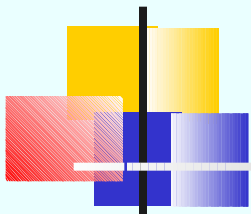


Ví dụ: đảo chuỗi – server (2)

```
// Chấp nhận kết nối và xử lý dữ liệu
public void listening(){
    byte[] receiveData = new byte[1024];
    byte[] sendData = new byte[1024];

    while(true){
        try {
            // Nhận dữ liệu
            DatagramPacket receivePacket = new
                DatagramPacket(receiveData,
                    receiveData.length);
            myServer.receive(receivePacket);
            input = new String(receivePacket.getData());

            // Xử lý dữ liệu
            ReverseString str = new ReverseString(input);
            str.reverse();
        }
    }
}
```



Ví dụ: đảo chuỗi – server (3)

```
// Dong goi thong tin du lieu can tra lai
```

```
InetAddress IPAddress =  
    receivePacket.getAddress();  
int port = receivePacket.getPort();  
sendData = str.get_string().getBytes();  
DatagramPacket sendPacket =  
    new DatagramPacket(sendData,  
        sendData.length, IPAddress, port);
```

```
// Gui du lieu ve client  
myServer.send(sendPacket);
```

```
} catch (SocketException e) {  
    System.out.println(e);  
} catch (IOException e) {  
    System.out.println(e);  
}
```

```
}
```

```
}  
}
```



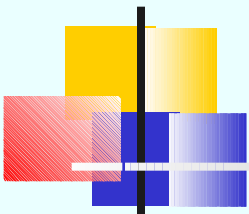
Ví dụ: đảo chuỗi – client (1)

```
import java.io.IOException;
import java.net.DatagramSocket;
import java.net.DatagramPacket;
import java.net.InetAddress;
import java.net.SocketException;

public class UDPClient {

    // khai bao socket cho client, cong gui va nhan
    // du lieu
    DatagramSocket mySocket = null;
    int port = 9980;

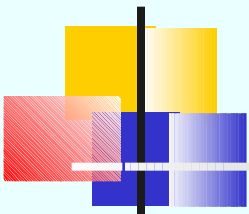
    // Tao ket noi
    public void connection(){
        try {
            mySocket = new DatagramSocket(port);
        } catch (SocketException e) {
            System.err.println(e);
        }
    }
}
```



Ví dụ: đảo chuỗi – client (2)

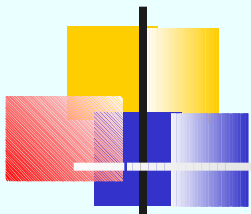
```
// gui du lieu den server
public void send(String str) {
    if (mySocket != null) {
        byte[] sendData = new byte[1024]; // bo dem gui dl
        try {
            InetAddress IPAddress =
                InetAddress.getByName("localhost");

            sendData = str.getBytes();
            DatagramPacket sendPacket =
                new DatagramPacket(sendData,
                    sendData.length, IPAddress, serverPort);
            mySocket.send(sendPacket);
        } catch (SocketException e) {
            System.err.println(e);
        } catch (IOException e) {
            System.err.println(e);
        }
    }
}
```



Ví dụ: đảo chuỗi – client (3)

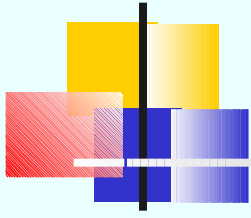
```
// nhan du lieu tra ve tu server
public String receive(){
    if (mySocket != null) {
        byte[] receiveData = new byte[1024]; // bo dem nhan dl
        try {
            DatagramPacket receivePacket =
                new DatagramPacket(receiveData,
                                    receiveData.length);
            mySocket.receive(receivePacket);
            return new String(receivePacket.getData());
        } catch (SocketException e) {
            System.err.println(e);
        } catch (IOException e) {
            System.err.println(e);
        }
    }
    return null;
}
```



Ví dụ: đảo chuỗi – client (4)

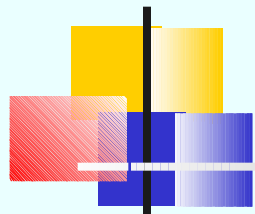
```
// dong cac ket noi
public void close(){
    if (mySocket != null ) {
        try {
            mySocket.close();

        } catch (Exception e)
        {
            System.err.println(e);
        }
    }
}
```



Bài tập (1)

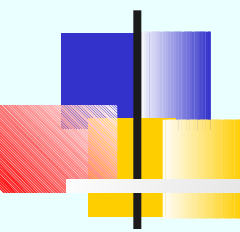
- Cài đặt theo mô hình giao thức UDP cho bài toán:
- Client yêu cầu người dùng nhập từ bàn phím hai số nguyên a và b
- server nhận và tính tổng a và b, sau đó trả về kết quả cho client
- Client nhận lại kết quả tổng và show ra màn hình cho người dùng



Bài tập (2)

Cùng yêu cầu, nhưng cài đặt theo mô hình MVC

- Cài đặt theo mô hình giao thức UDP cho bài toán:
- Client yêu cầu người dùng nhập từ bàn phím hai số nguyên a và b
- server nhận và tính tổng a và b , sau đó trả về kết quả cho client
- Client nhận lại kết quả tổng và show ra màn hình cho người dùng



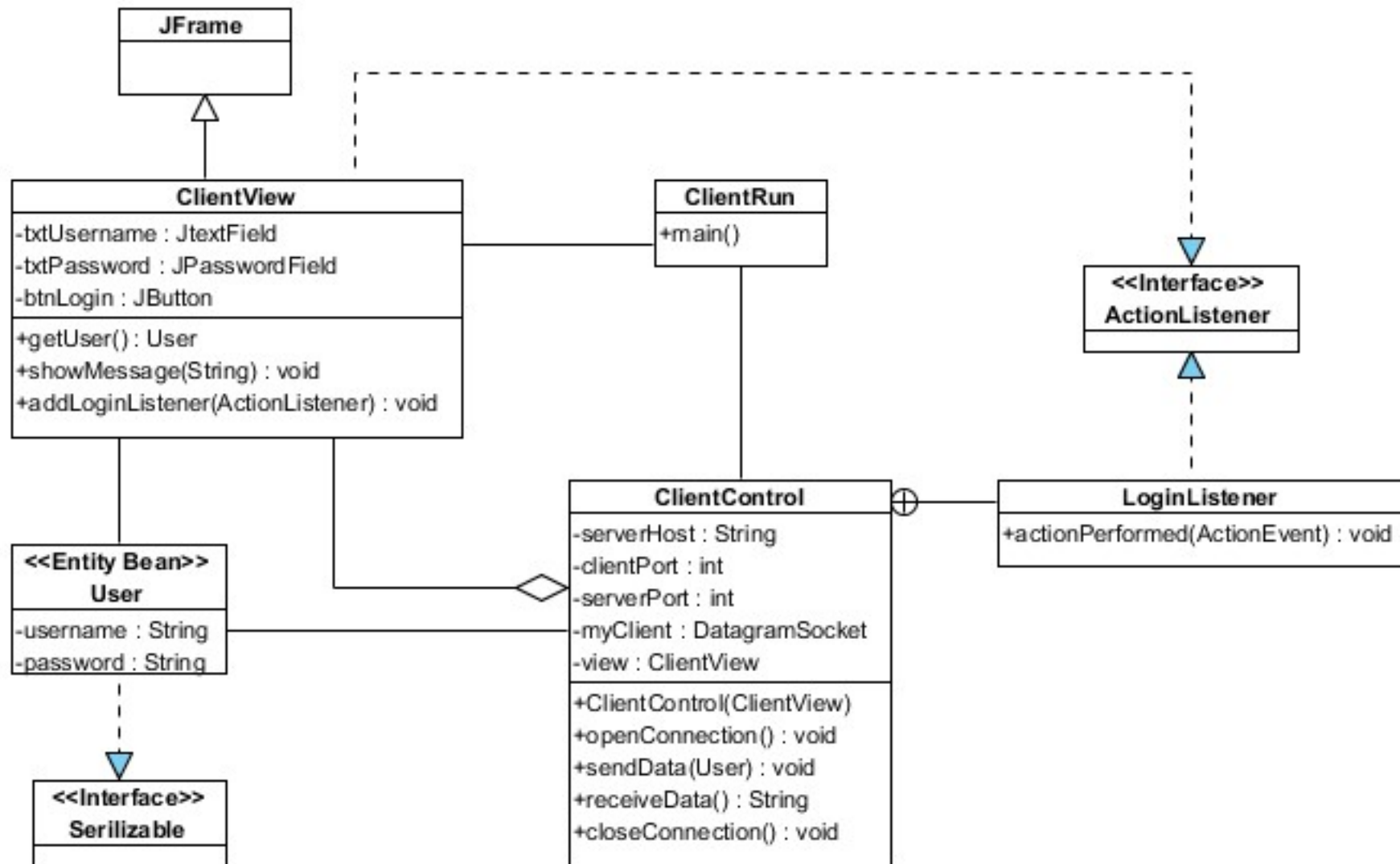
Ví dụ: Login từ xa dùng UDP



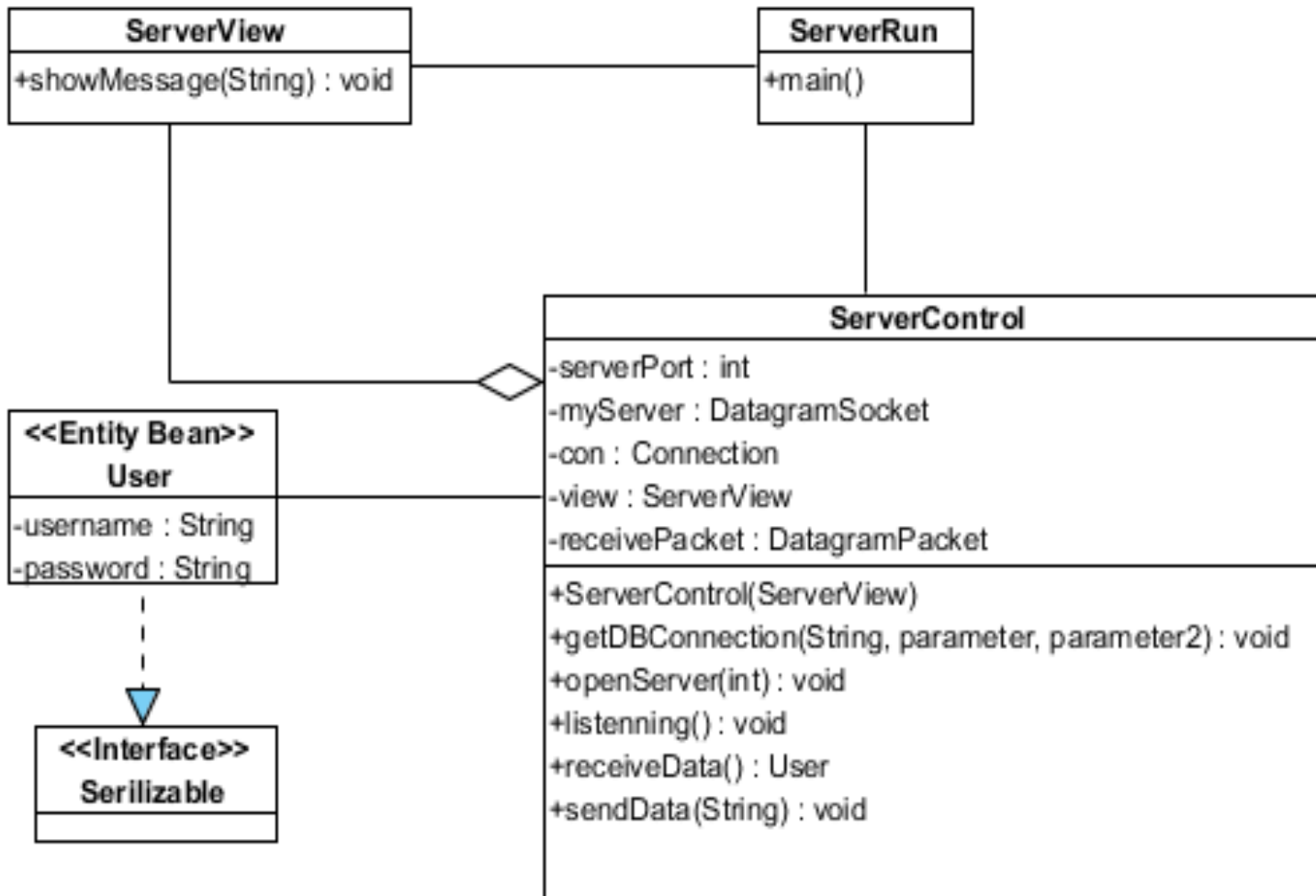
Bài toán: Login dùng UDP

- Thông tin user được lưu trên server UDP
- Chương trình hiện cửa sổ đăng nhập GUI (username, password) ở phía client UDP
- Khi click vào nút login, client sẽ gửi thông tin đăng nhập lên server để xử lý
- Kết quả đăng nhập được trả từ server về client và client thông báo lại cho người dùng

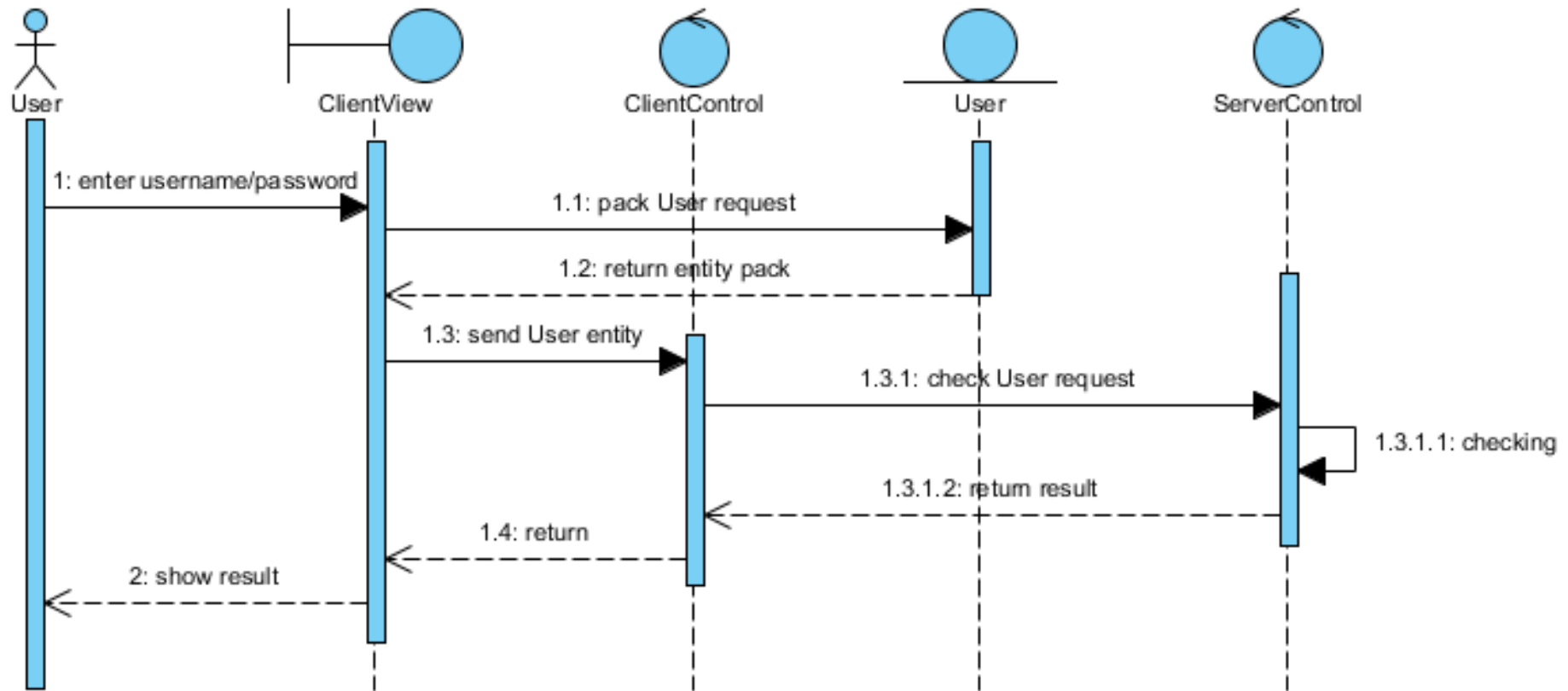
Sơ đồ lớp phía client



Sơ đồ lớp phía server



Tuần tự thực hiện





Lớp: User

```
import java.io.Serializable;

public class User implements Serializable{
    private String userName;
    private String password;

    public User(){
    }

    public User(String username, String password){
        this.userName = username;
        this.password = password;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public String getUserName() {
        return userName;
    }

    public void setUserName(String userName) {
        this.userName = userName;
    }
}
```

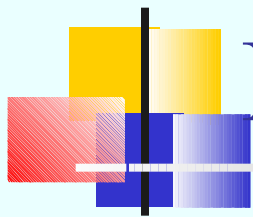


Lớp: ClientView (1)

```
import java.awt.FlowLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JPasswordField;
import javax.swing.JTextField;

public class ClientView extends JFrame implements ActionListener{
    private JTextField txtUsername;
    private JPasswordField txtPassword;
    private JButton btnLogin;
```

Lớp: ClientView (2)

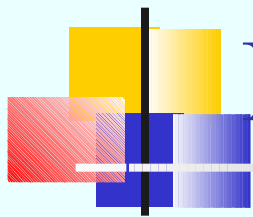
```
public ClientView(){
    super("UDP Login MVC");

    txtUsername = new JTextField(15);
    txtPassword = new JPasswordField(15);
    txtPassword.setEchoChar('*');
    btnLogin = new JButton("Login");

    JPanel content = new JPanel();
    content.setLayout(new FlowLayout());
    content.add(new JLabel("Username:"));
    content.add(txtUsername);
    content.add(new JLabel("Password:"));
    content.add(txtPassword);
    content.add(btnLogin);

    this.setContentPane(content);
    this.pack();

    this.addWindowListener(new WindowAdapter(){
        public void windowClosing(WindowEvent e){
            System.exit(0);
        }
    });
}
```



Lớp: ClientView (3)

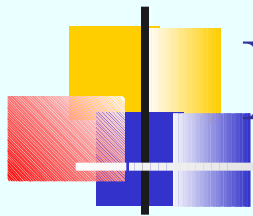
```
public void actionPerformed(ActionEvent e) {  
}  
  
public User getUser(){  
    User model = new User(txtUsername.getText(),  
        txtPassword.getText());  
    return model;  
}  
  
public void showMessage(String msg){  
    JOptionPane.showMessageDialog(this, msg);  
}  
  
public void addLoginListener(ActionListener log) {  
    btnLogin.addActionListener(log);  
}  
}
```



Lớp: ClientControl (1)

```
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;

public class ClientControl {
    private ClientView view;
    private int serverPort = 5555;
    private int clientPort = 6666;
    private String serverHost = "localhost";
    private DatagramSocket myClient;
```



Lớp: ClientControl (2)

```
public ClientControl(ClientView view){
    this.view = view;
    this.view.addLoginListener(new LoginListener());
}

class LoginListener implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        openConnection();

        User user = view.getUser();
        sendData(user);

        String result = receiveData();
        if(result.equals("ok"))
            view.showMessage("Login succesfully!");
        else
            view.showMessage("Invalid username and/or
password!");

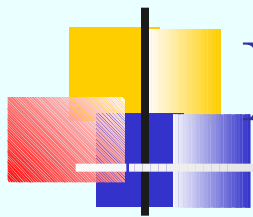
        closeConnection();
    }
}
```



Lớp: ClientControl (3)

```
private void openConnection(){
    try {
        myClient = new DatagramSocket(clientPort);
    } catch (Exception ex) {
        view.showMessage(ex.getStackTrace().toString());
    }
}

private void closeConnection(){
    try {
        myClient.close();
    } catch (Exception ex) {
        view.showMessage(ex.getStackTrace().toString());
    }
}
```

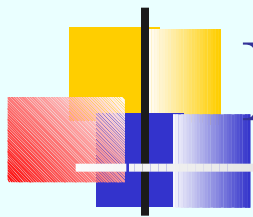


Lớp: ClientControl (4)

```
private void sendData(User user){
    try {
        ByteArrayOutputStream baos = new ByteArrayOutputStream();
        ObjectOutputStream oos = new ObjectOutputStream(baos);
        oos.writeObject(user);
        oos.flush();

        InetAddress IPAddress =
            InetAddress.getByName(serverHost);
        byte[] sendData = baos.toByteArray();
        DatagramPacket sendPacket = new DatagramPacket(sendData,
            sendData.length, IPAddress, serverPort);
        myClient.send(sendPacket);

    } catch (Exception ex) {
        view.showMessageDialog(ex.getStackTrace().toString());
    }
}
```



Lớp: ClientControl (5)

```
private String receiveData(){
    String result = "";
    try {
        byte[] receiveData = new byte[1024];
        DatagramPacket receivePacket = new
            DatagramPacket(receiveData, receiveData.length);
        myClient.receive(receivePacket);

        ByteArrayInputStream bais = new
            ByteArrayInputStream(receiveData);
        ObjectInputStream ois = new ObjectInputStream(bais);
        result = (String)ois.readObject();
    } catch (Exception ex) {
        view.showMessageDialog(ex.getStackTrace().toString());
    }
    return result;
}
```



Lớp: ClientRun

```
public class ClientRun {  
  
    public static void main(String[] args) {  
        ClientView view = new ClientView();  
        ClientControl control = new ClientControl(view);  
        view.setVisible(true);  
    }  
}
```




Lớp: ServerView

```
public class ServerView {  
    public ServerView(){  
    }  
  
    public void showMessage(String msg){  
        System.out.println(msg);  
    }  
}
```



Lớp: ServerControl (1)

```
import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
import udp.client.User;

public class ServerControl {
    private ServerView view;
    private Connection con;
    private DatagramSocket myServer;
    private int serverPort = 5555;
    private DatagramPacket receivePacket = null;
```



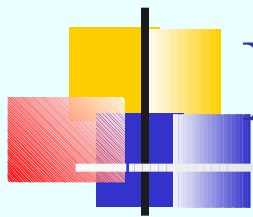
Lớp: ServerControl (2)

```
public ServerControl(ServerView view){
    this.view = view;
    getDBConnection("myDBName", "admin", "123456");
    openServer(serverPort);
    view.showMessage("UDP server is running...");

    while(true){
        listenning();
    }
}

private void getDBConnection(String dbName, String username, String
password){
    String dbUrl = "jdbc:mysql://your.database.domain/" + dbName;
    String dbClass = "com.mysql.jdbc.Driver";

    try {
        Class.forName(dbClass);
        con = DriverManager.getConnection (dbUrl, username, password);
    }catch(Exception e) {
        view.showMessage(e.getStackTrace().toString());
    }
}
```



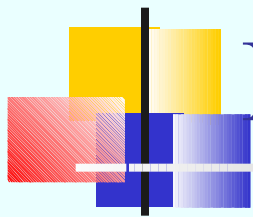
Lớp: ServerControl (3)

```
private void openServer(int portNumber){
    try {
        myServer = new DatagramSocket(portNumber);
    } catch (IOException e) {
        view.showMessage(e.toString());
    }
}

private void listenning(){
    User user = receiveData();

    String result = "false";
    if(checkUser(user)){
        result = "ok";
    }

    sendData(result);
}
```

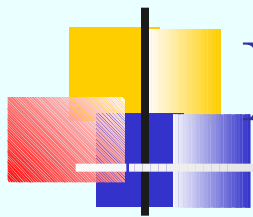


Lớp: ServerControl (3)

```
private void sendData(String result){
    try {
        ByteArrayOutputStream baos = new ByteArrayOutputStream();
        ObjectOutputStream oos = new ObjectOutputStream(baos);
        oos.writeObject(result);
        oos.flush();

        InetAddress IPAddress = receivePacket.getAddress();
        int clientPort = receivePacket.getPort();
        byte[] sendData = baos.toByteArray();
        DatagramPacket sendPacket = new DatagramPacket(sendData,
            sendData.length, IPAddress, clientPort);
        myServer.send(sendPacket);

    } catch (Exception ex) {
        view.showMessageDialog(ex.getStackTrace().toString());
    }
}
```

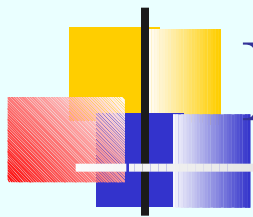


Lớp: ServerControl (4)

```
private User receiveData(){
    User user = null;
    try {
        byte[] receiveData = new byte[1024];
        receivePacket = new
            DatagramPacket(receiveData, receiveData.length);
        myServer.receive(receivePacket);

        ByteArrayInputStream bais = new
            ByteArrayInputStream(receiveData);
        ObjectInputStream ois = new ObjectInputStream(bais);
        user = (User)ois.readObject();

    } catch (Exception ex) {
        view.showMessageDialog(ex.getStackTrace().toString());
    }
    return user;
}
```



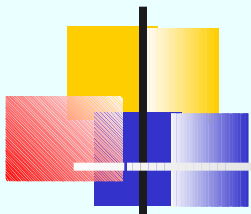
Lớp: ServerControl (5)

```
private boolean checkUser(User user) {  
    String query = "Select * FROM users WHERE username='"  
        + user.getUserName()  
        + "' AND password='" + user.getPassword() + "'";  
  
    try {  
        Statement stmt = con.createStatement();  
        ResultSet rs = stmt.executeQuery(query);  
  
        if (rs.next()) {  
            return true;  
        }  
    } catch (Exception e) {  
        view.showMessageDialog(e.getStackTrace().toString());  
    }  
    return false;  
}
```



Lớp: ServerRun

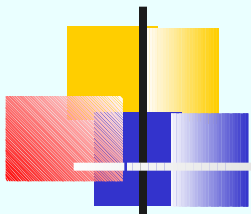
```
public class ServerRun {  
    public static void main(String[] args) {  
        ServerView view      = new ServerView();  
        ServerControl control = new ServerControl(view);  
    }  
}
```

Bài tập (1)

Cài đặt đúng mô hình MVC (cũ hoặc mới) cho bài toán quản lí người dùng theo UDP:

- Server chứa CSDL về người dùng, có bảng tbluser chứa các cột: id, username, password, address, birthday, sex, description
- Client có giao diện nhập thông tin đăng kí người dùng mới
- Sau khi nhập thông tin và click submit, client gửi thông tin đăng kí đến server
- Server kiểm tra xem có trùng username không, nếu không thì thêm vào CSDL và báo thành công, nếu trùng thì thông báo trùng cho client
- Client nhận được thông tin sẽ hiển thị yêu cầu người dùng nhập lại khi trùng, hoặc báo đăng kí thành công.



Bài tập (2)

Cài đặt đúng mô hình MVC (cũ hoặc mới) cho bài toán quản lí người dùng theo UDP:

- Server chứa CSDL về người dùng, có bảng tbluser chứa các cột: id, username, password, address, birthday, sex, description
- Client có giao diện nhập thông tin tìm kiếm người dùng theo tên
- Sau khi nhập thông tin và click submit, client gửi thông tin tìm kiếm đến server
- Server tìm kiếm thông tin người dùng từ CSDL và trả kết quả về cho client
- Client nhận được thông tin sẽ hiển thị danh sách người dùng có tên chứa từ khóa đã nhập.



Questions?
