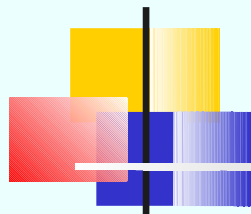


# Lập trình mạng

# Lập trình Socket với TCP/IP

---

*Giảng viên: TS. Nguyễn Mạnh Hùng*  
*Học viện Công nghệ Bưu chính Viễn thông (PTIT)*

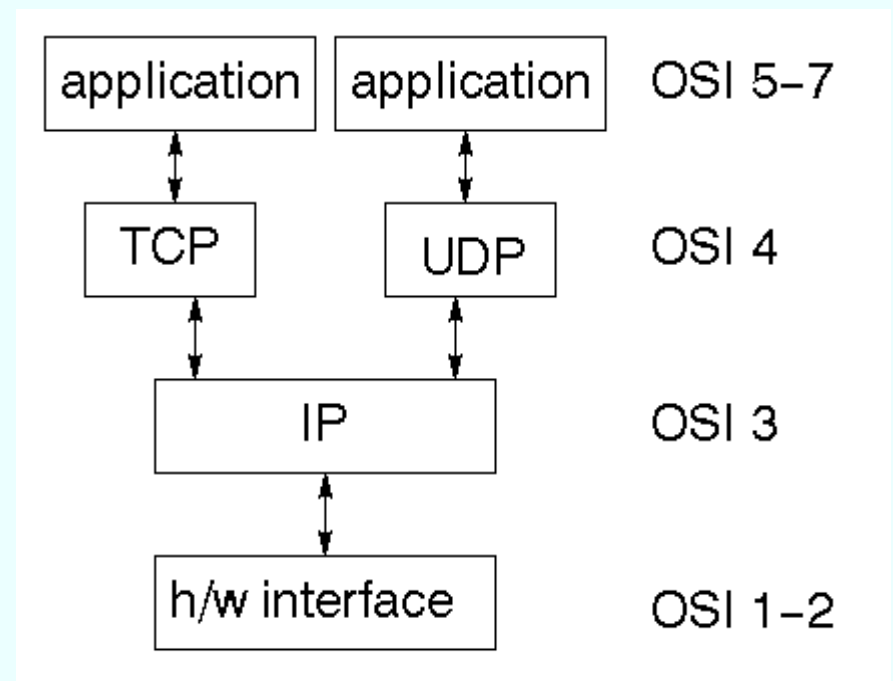
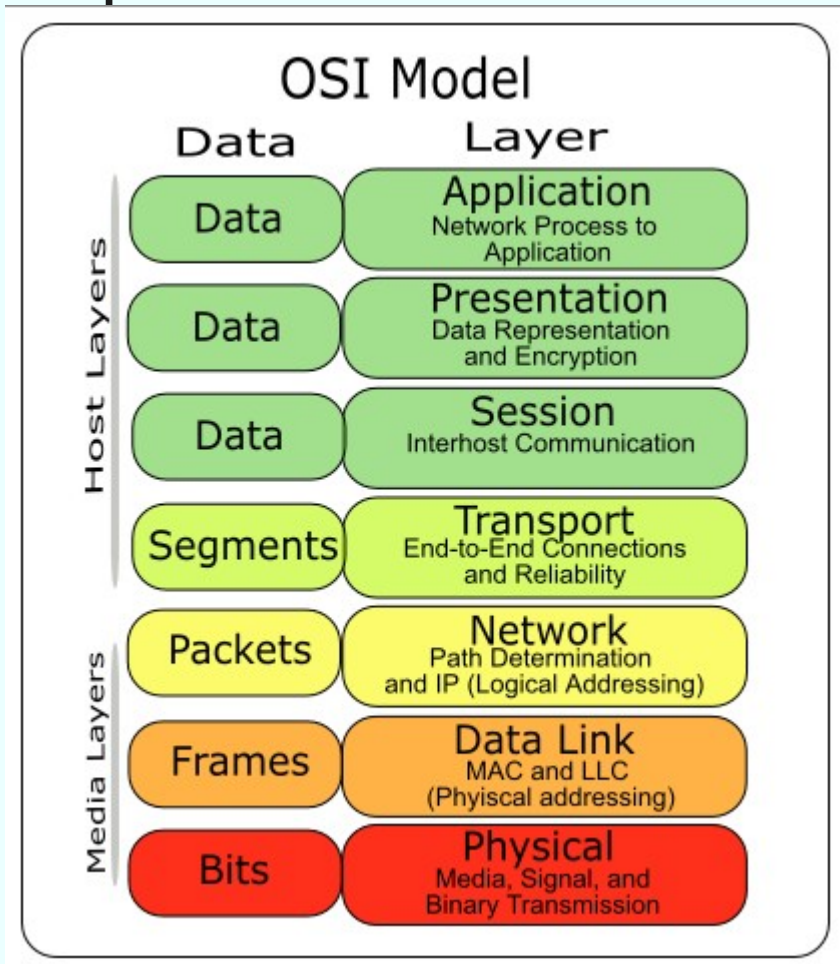


# Nội dung

---

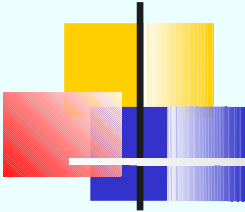
- Giao thức TCP/IP
- Cài đặt phía server
- Cài đặt phía client
- Ví dụ: đảo ngược chuỗi
- Bài tập

# TCP/IP trong mô hình ISO



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

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

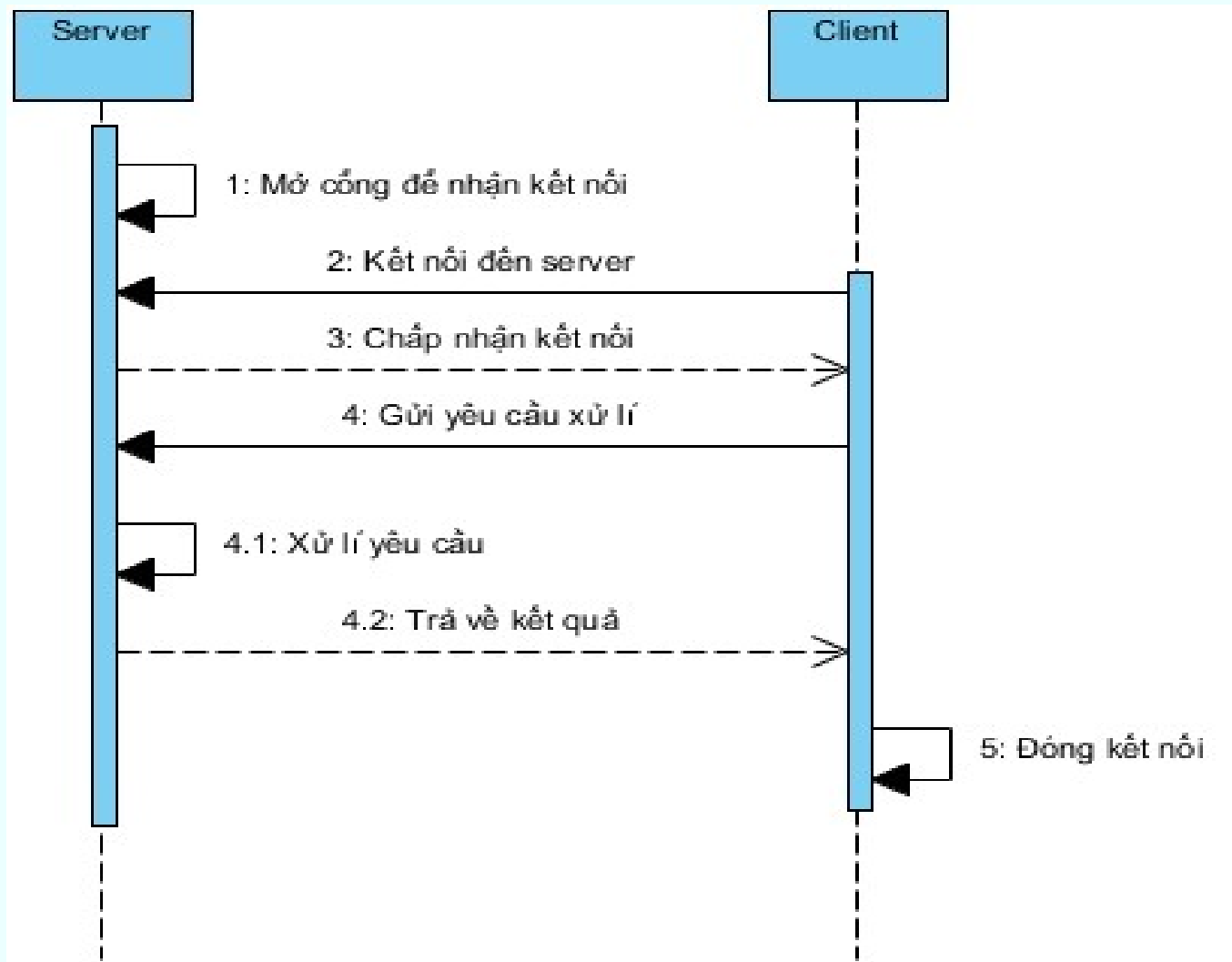


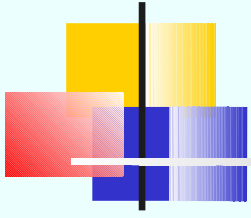
# Cấu trúc gói tin TCP/IP

0	4	8	16	19	31
Version	IHL	Type of Service	Total Length		
Identification			Flags	Fragment Offset	
Time To Live		Protocol	Header Checksum		
Source IP Address					
Destination IP Address					
Options					Padding

[image source: <http://www.freesoft.org>]

# Giao thức TCP/IP



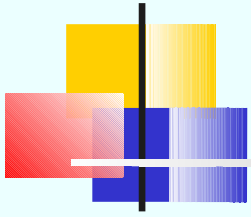


# Server (1)

---

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

```
try {  
    ServerSocket myServer = new ServerSocket(số cổng);  
}  
catch (IOException e) {  
    System.out.println(e);  
}
```

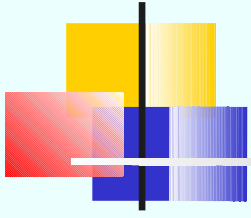


# Server (2)

---

Bước 2: Tạo một đối tượng socket từ ServerSocket để lắng nghe và chấp nhận các kết nối từ phía client

```
try {  
    Socket clientSocket = myServer.accept();  
  
    DataInputStream is = new  
        DataInputStream(clientSocket.getInputStream());  
    PrintStream os = new  
        PrintStream(clientSocket.getOutputStream());  
} catch (IOException e) {  
    System.out.println(e);  
}
```



# Server (3)

---

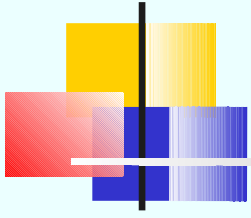
Bước 3: Mỗi khi nhận được dữ liệu từ client, tiến hành xử lý và gửi trả về client đó

```
// Xu li du lieu nhan duoc va tra ve
while (true) {
    // doc du lieu vao
    String input = is.read();

    // xu li du lieu
    ...

    // tra ve du lieu
    os.println(dữ liệu trả về);
}
```



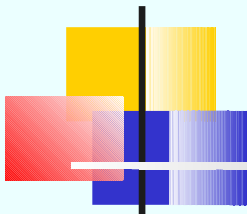


# Client (1)

---

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

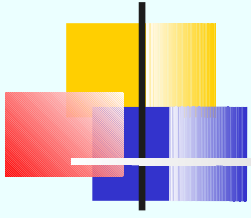
```
try {  
    Socket mySocket = new Socket(tên máy chủ, số cổng);  
} catch (UnknownHostException e) {  
    System.err.println(e);  
} catch (IOException e) {  
    System.err.println(e);  
}
```



# Client (2)

Bước 2: Mở luồng kết nối vào (nhận dữ liệu) và kết nối ra (gửi dữ liệu) đến socket vừa mở

```
try {  
    DataOutputStream os = new  
        DataOutputStream(mySocket.getOutputStream());  
    DataInputStream is = new  
        DataInputStream(mySocket.getInputStream());  
} catch (UnknownHostException e) {  
    System.err.println(e);  
} catch (IOException e) {  
    System.err.println(e);  
}
```

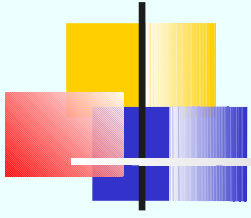


# Client (3)

---

## Bước 3: Gửi dữ liệu đến server

```
try {  
    os.writeBytes(dữ liệu gửi đi);  
  
} catch (UnknownHostException e) {  
    System.err.println("e");  
} catch (IOException e) {  
    System.err.println("e");  
}
```

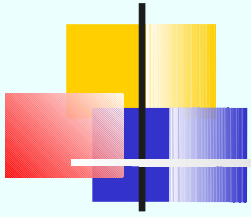


# Client (4)

---

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

```
try {  
    String responseStr; // du lieu nhan ve  
    if ((responseStr = is.read()) != null) {  
        return responseStr;  
    }  
} catch (UnknownHostException e) {  
    System.err.println(e);  
} catch (IOException e) {  
    System.err.println(e);  
}
```

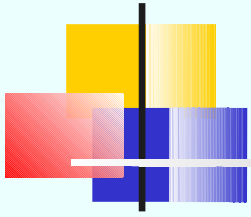


# Client (5)

---

## Bước 5: Đóng các kết nối tới server

```
try {  
    os.close();  
    is.close();  
    mySocket.close();  
} catch (UnknownHostException e) {  
    System.err.println(e);  
} catch (IOException e) {  
    System.err.println(e);  
}
```



# Ví dụ: đảo chuỗi (1)

```
import java.lang.String;

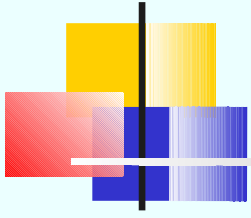
public class ReverseString {
    private String _string;

    // khoi tao khong tham so
    public ReverseString() {}

    // khoi tao co tham so
    public ReverseString(String _string) {
        this._string = _string;
    }

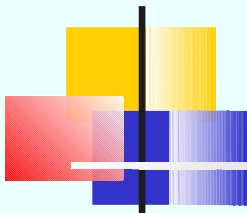
    public String get_string() {
        return _string;
    }

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



# Ví dụ: đảo chuỗi (2)

```
//phuong thuc dao nguoc chuoì ki 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.io.DataInputStream;
import java.io.IOException;
import java.io.PrintStream;
import java.net.ServerSocket;
import java.net.Socket;

public class TCPServer {
    // Khai bao server socket, luong vao-ra, va doi tuong socket
    ServerSocket myServer = null;
    String input;
    DataInputStream is;
    PrintStream os;
    Socket clientSocket = null;

    // Mo mot server socket
    public void openServer() {
        try {
            myServer = new ServerSocket(9999);
        } catch (IOException e) {
            System.out.println(e);
        }
    }
}
```





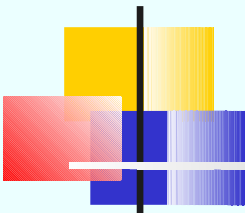
# Ví dụ: đảo chuỗi – server (2)

```
// Chấp nhận kết nối và xử lý dữ liệu
public void listening() {
    try {
        clientSocket = myServer.accept();
        is = new DataInputStream(clientSocket.getInputStream());
        os = new PrintStream(clientSocket.getOutputStream());

        // Xử lý dữ liệu nhận được và trả về
        while (true) {
            // đọc dữ liệu vào
            input = is.readLine();

            // xử lý dữ liệu
            ReverseString str = new ReverseString(input);
            str.reverse();

            // trả về dữ liệu
            os.println(str.get_string());
        }
    } catch (IOException e) {
        System.out.println(e);
    }
}
```



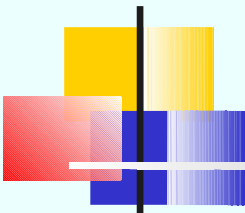
# Ví dụ: đảo chuỗi – client (1)

```
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.net.Socket;
import java.net.UnknownHostException;

public class TCPClient {

    // khai bao socket cho client, luong vao-ra
    Socket mySocket = null;
    DataOutputStream os = null;
    DataInputStream is = null;

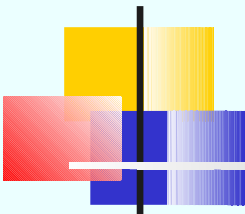
    // Tao ket noi
    public void connection(){
        try {
            mySocket = new Socket("hostname", 9999);
            os = new DataOutputStream(mySocket.getOutputStream());
            is = new DataInputStream(mySocket.getInputStream());
        } catch (UnknownHostException e) {
            System.err.println(e);
        } catch (IOException e) {
            System.err.println(e);
        }
    }
}
```



# Ví dụ: đảo chuỗi – client (2)

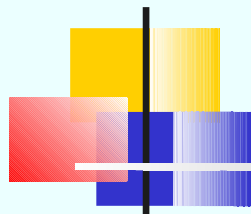
```
public void send(String str){ // gui du lieu den server
    if (mySocket != null && os != null) {
        try {
            os.writeBytes(str);
        } catch (UnknownHostException e) {
            System.err.println(e);
        } catch (IOException e) {
            System.err.println(e);
        }
    }
}

public String receive(){ // nhan du lieu tra ve tu server
    if (mySocket != null && is != null) {
        try {
            String responseStr;
            if ((responseStr = is.readLine()) != null) {
                return responseStr;
            }
        } catch (UnknownHostException e) {
            System.err.println(e);
        } catch (IOException e) {
            System.err.println(e);
        }
    }
    return null;
}
```



# Ví dụ: đảo chuỗi – client (3)

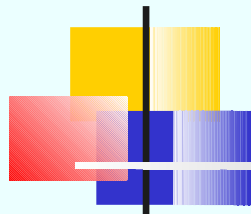
```
// đóng các kết nối
public void close() {
    if (mySocket != null && os != null && is != null) {
        try {
            os.close();
            is.close();
            mySocket.close();
        } catch (UnknownHostException e) {
            System.err.println(e);
        } catch (IOException e) {
            System.err.println(e);
        }
    }
}
```



# Bài tập (1)

---

- Cài đặt theo mô hình giao thức TCI/IP 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

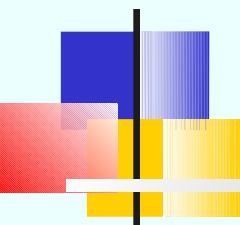


## Bài tập (2)

---

Cùng yêu cầu, nhưng cài đặt đúng mô hình MVC

- Cài đặt theo mô hình giao thức TCI/IP 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 TCP/IP

---



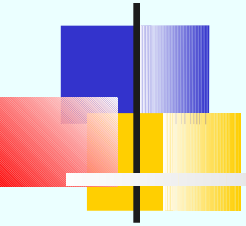
# Bài toán: Login dùng TCP/IP

---

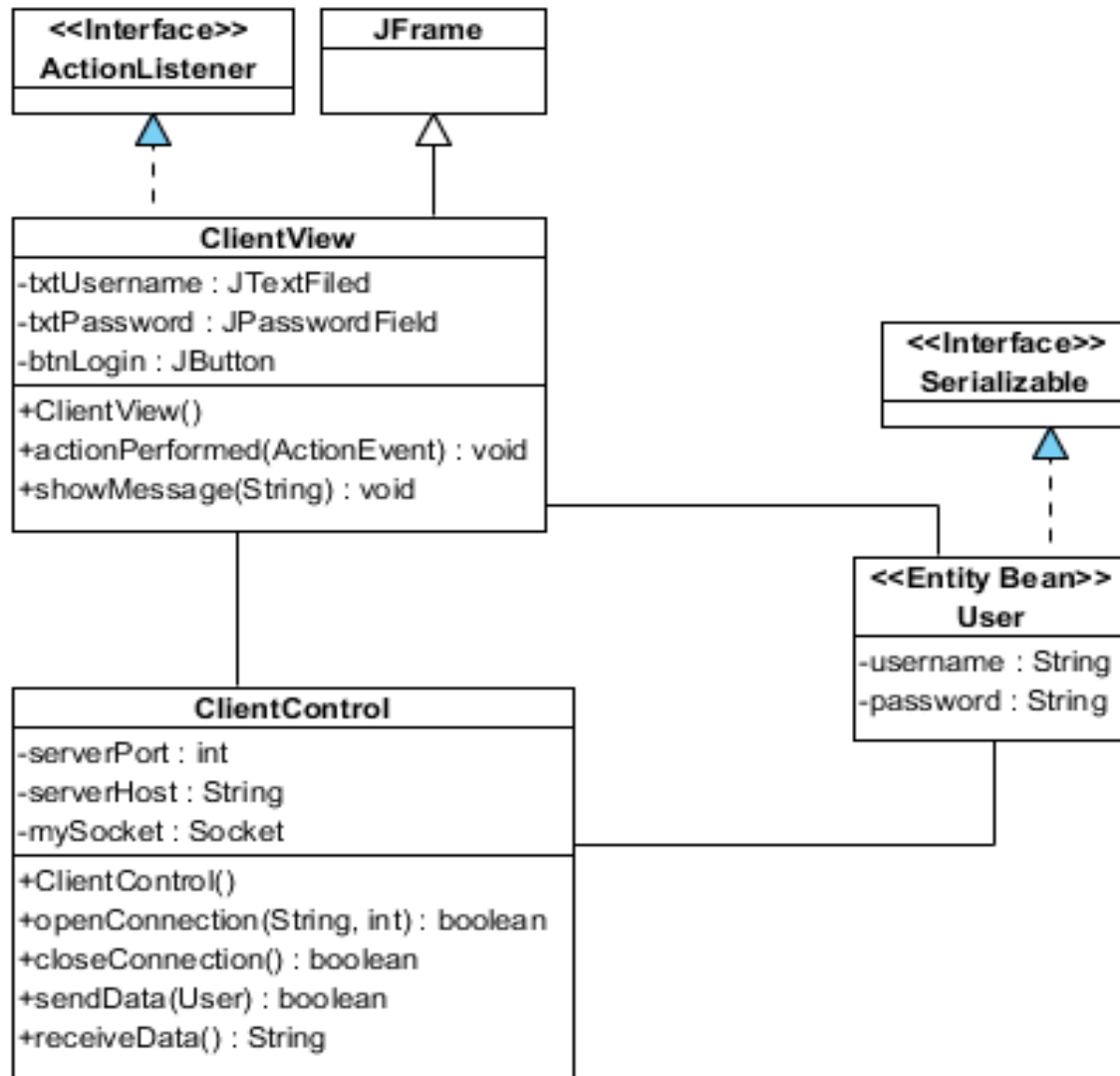
- Thông tin user được lưu trên server TCP
- Chương trình hiện cửa sổ đăng nhập GUI (username, password) ở phía client TCP
- Khi click vào nút login, client sẽ gửi thông tin đăng nhập lên server để xử lý
- Kết quả đăng nhập được trả từ server về client và client thông báo lại cho người dùng



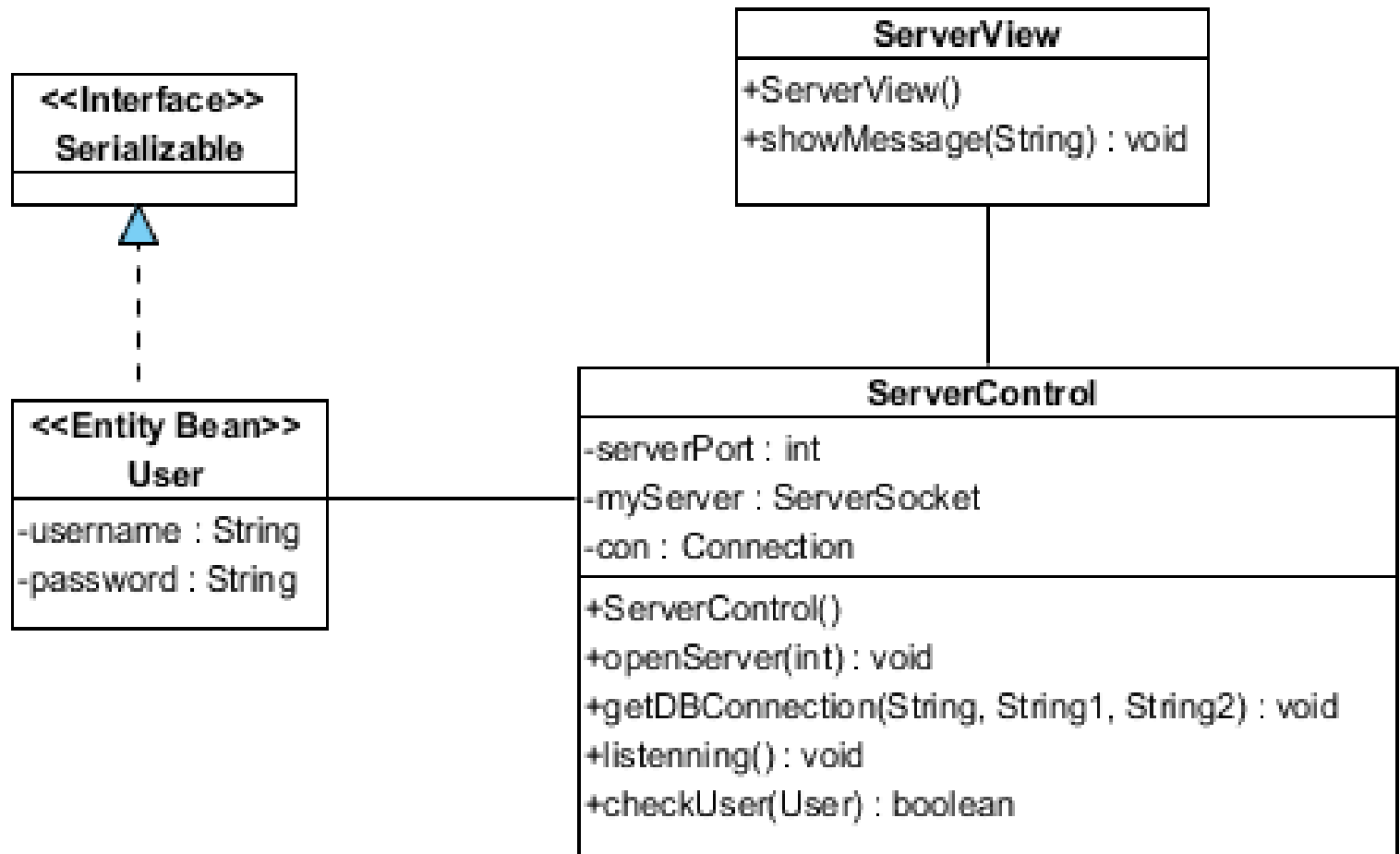
# Cài đặt theo mô hình MVC cổ điển



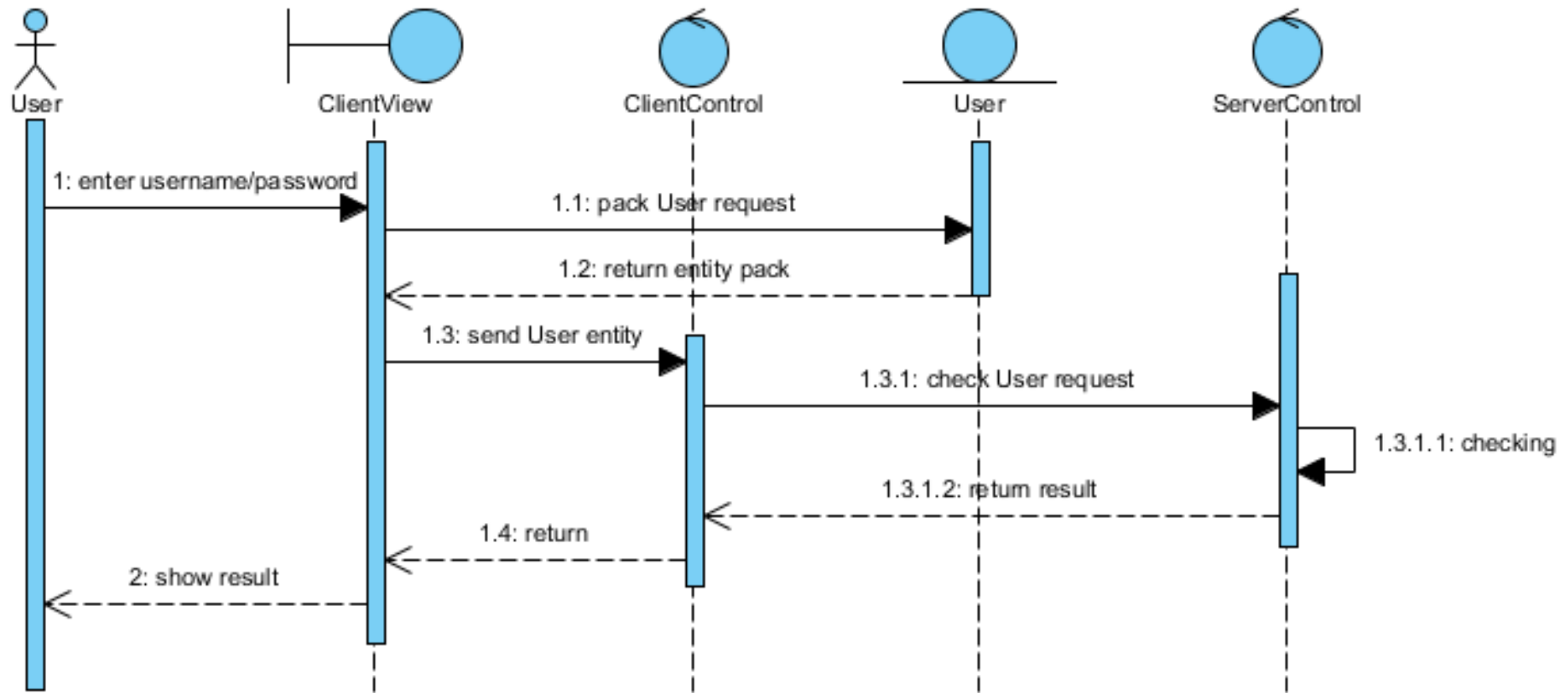
# Sơ đồ lớp phía client



# 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("TCP 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) {
    if(e.getSource().equals(btnLogin)){
        User model = new User(txtUsername.getText(),
                               txtPassword.getText());
        ClientControl clientCtr = new ClientControl();
        clientCtr.openConnection();
        clientCtr.sendData(model);

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

public void showMessage(String msg){
    JOptionPane.showMessageDialog(this, msg);
}
```





# Lớp: ClientControl (1)

---

```
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.net.Socket;

public class ClientControl {
    private Socket mySocket;
    private String serverHost = "localhost";
    private int serverPort = 8888;

    public ClientControl(){
    }
```



# Lớp: ClientControl (2)

---

```
public Socket openConnection(){
    try {
        mySocket = new Socket(serverHost, serverPort);
    } catch (Exception ex) {
        ex.printStackTrace();
        return null;
    }
    return mySocket;
}

public boolean sendData(User user){
    try {
        ObjectOutputStream oos =
            new ObjectOutputStream(mySocket.getOutputStream());
        oos.writeObject(user);

    } catch (Exception ex) {
        ex.printStackTrace();
        return false;
    }
    return true;
}
```



# Lớp: ClientControl (3)

---

```
public String receiveData(){
    String result = null;
    try {
        ObjectInputStream ois =
            new ObjectInputStream(mySocket.getInputStream());
        Object o = ois.readObject();
        if(o instanceof String){
            result = (String)o;
        }
    } catch (Exception ex) {
        ex.printStackTrace();        return null;
    }
    return result;
}
```

```
public boolean closeConnection(){
    try {
        mySocket.close();
    } catch (Exception ex) {
        ex.printStackTrace();        return false;
    }
    return true;
}
```



# Lớp: ClientRun

---

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



# Lớp: ServerView

---

```
public class ServerView {  
    public ServerView(){  
        new ServerControl();  
        showMessage("TCP server is running...");  
    }  
  
    public void showMessage(String msg){  
        System.out.println(msg);  
    }  
}
```



# Lớp: ServerControl (1)

---

```
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.net.ServerSocket;
import java.net.Socket;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
import tcp.client.User;

public class ServerControl {
    private Connection con;
    private ServerSocket myServer;
    private int serverPort = 8888;

    public ServerControl(){
        getDBConnection("myDBName", "admin", "123456");
        openServer(serverPort);

        while(true){
            listenning();
        }
    }
}
```



# Lớp: ServerControl (2)

---

```
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) {
        e.printStackTrace();
    }
}

private void openServer(int portNumber){
    try {
        myServer = new ServerSocket(portNumber);
    }catch(IOException e) {
        e.printStackTrace();
    }
}
```



# Lớp: ServerControl (3)

---

```
private void listenning(){
    try {
        Socket clientSocket = myServer.accept();
        ObjectInputStream ois = new
            ObjectInputStream(clientSocket.getInputStream());
        ObjectOutputStream oos = new
            ObjectOutputStream(clientSocket.getOutputStream());

        Object o = ois.readObject();
        if(o instanceof User){
            User user = (User)o;
            if(checkUser(user)){
                oos.writeObject("ok");
            }
            else
                oos.writeObject("false");
        }
    }catch (Exception e) {
        e.printStackTrace();
    }
}
```





# Lớp: ServerControl (4)

---

```
private boolean checkUser(User user) throws Exception {
    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) {
        throw e;
    }
    return false;
}
```

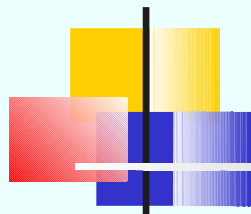


# Lớp: ServerRun

---

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

Lưu ý: chạy serverRun trước rồi chạy clientRun sau!

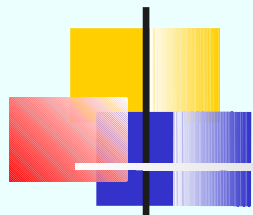


# Bài tập (1)

---

Cài đặt đúng mô hình MVC cổ điển cho bài toán quản lí người dùng theo TCP/IP:

- 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.

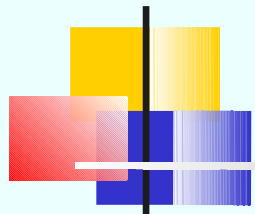


# Bài tập (2)

---

Cài đặt đúng mô hình MVC cổ điển cho bài toán quản lí người dùng theo TCP/IP:

- 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.



# Bài tập (3)

---

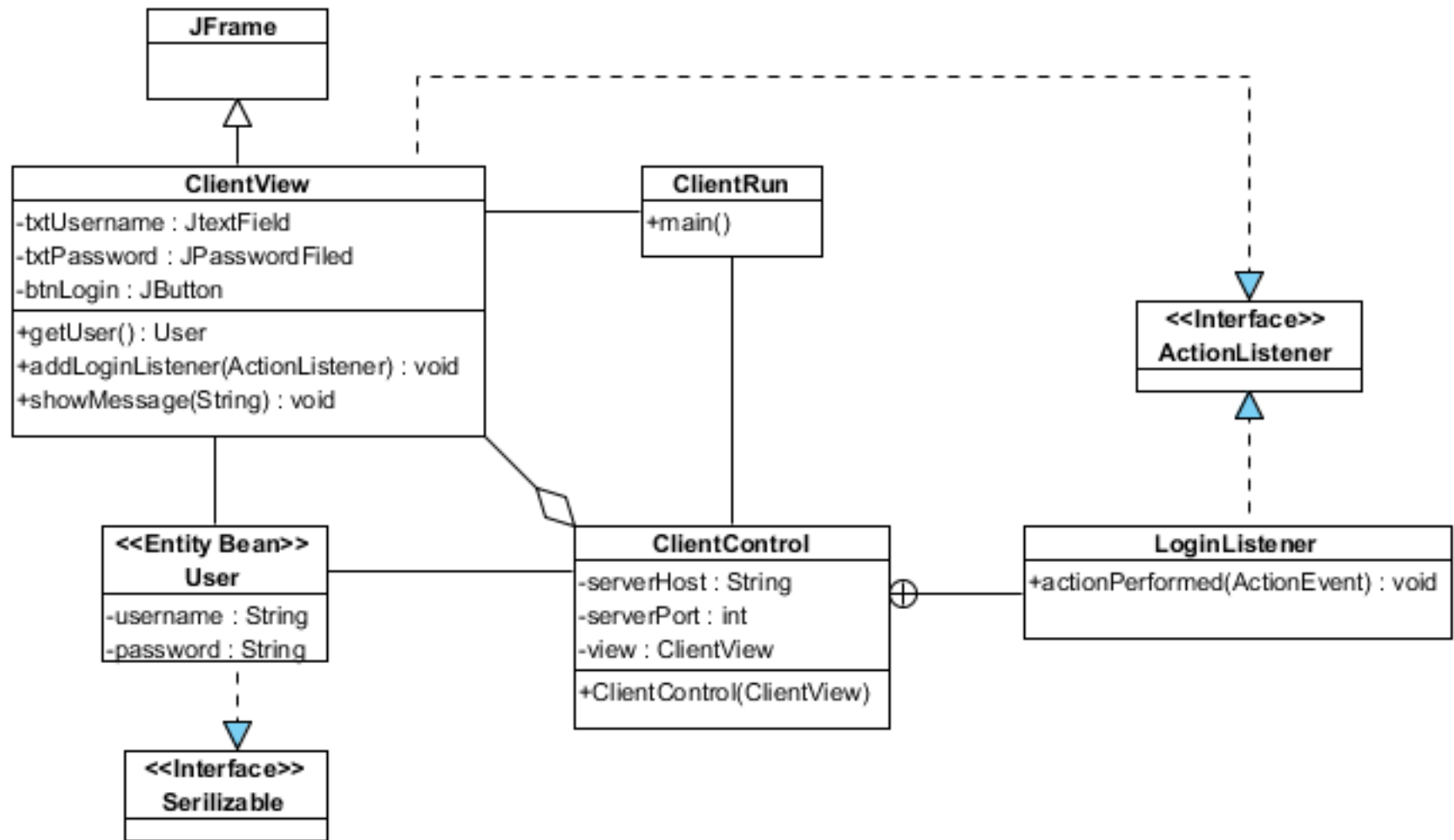
Cài đặt đúng mô hình MVC cổ điển cho bài toán quản lí người dùng theo TCP/IP sao cho Server có khả năng xử lí song song nhiều client kết nối đến đồng thời.



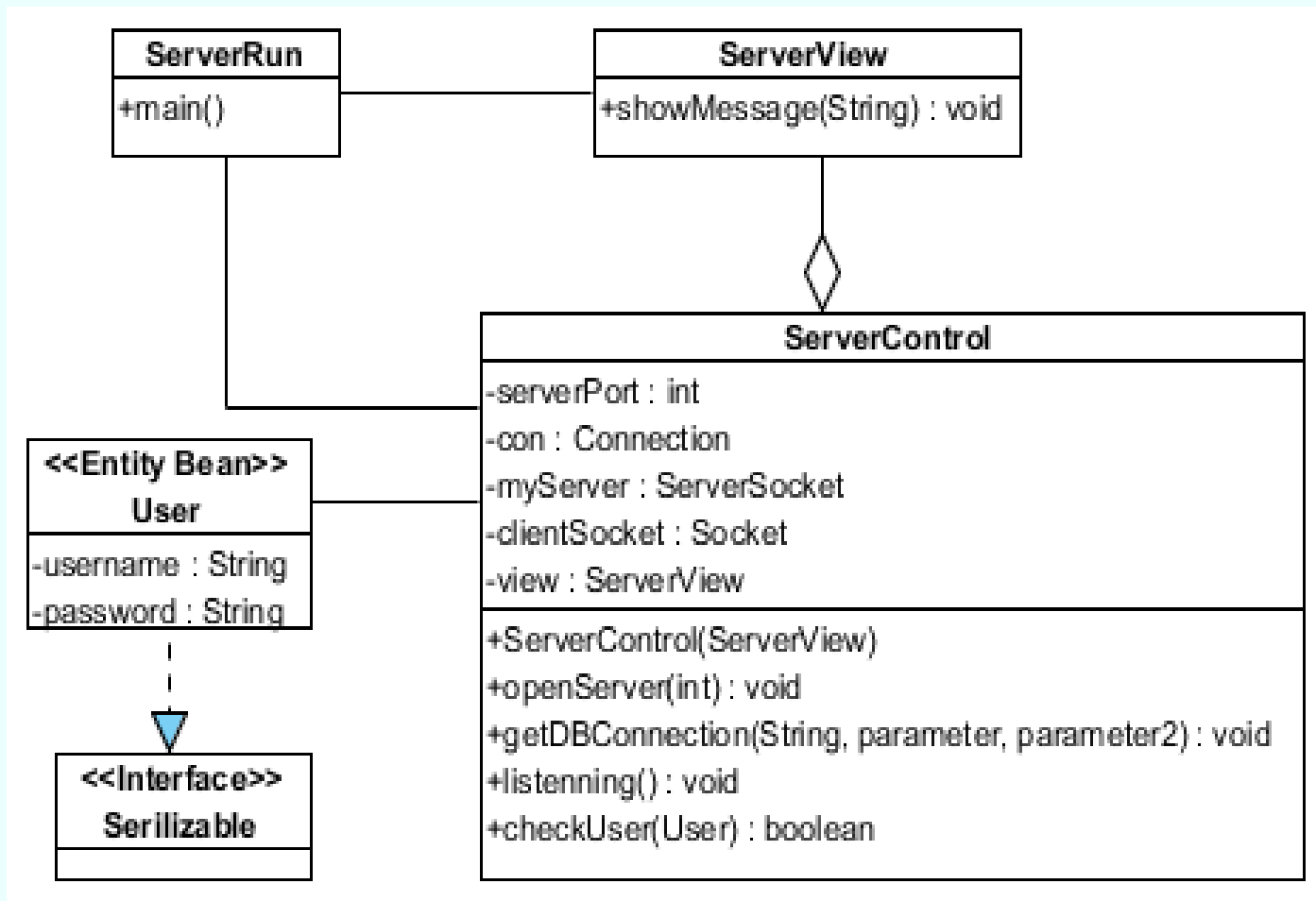
# Cài đặt theo mô hình MVC cải tiến

---

# Sơ đồ lớp phía client



# Sơ đồ lớp phía server







# 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("TCP 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.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.net.Socket;

public class ClientControl {
    private ClientView view;
    private String serverHost = "localhost";
    private int serverPort = 8888;

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



# Lớp: ClientControl (2)

---

```
class LoginListener implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        try {
            User user = view.getUser();
            Socket mySocket = new Socket(serverHost, serverPort);
            ObjectOutputStream oos = new
                ObjectOutputStream(mySocket.getOutputStream());
            oos.writeObject(user);

            ObjectInputStream ois = new
                ObjectInputStream(mySocket.getInputStream());
            Object o = ois.readObject();
            if(o instanceof String){
                String result = (String)o;
                if(result.equals("ok"))
                    view.showMessageDialog("Login succesfully!");
                else view.showMessageDialog("Invalid username and/or
password!");
            }
            mySocket.close();
        } catch (Exception ex) {
            view.showMessageDialog(ex.getStackTrace().toString());
        }
    }
}
```



# 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.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.net.ServerSocket;
import java.net.Socket;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
import tcp.client.User;

public class ServerControl {
    private ServerView view;
    private Connection con;
    private ServerSocket myServer;
    private Socket clientSocket;
    private int serverPort = 8888;

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

        while(true){
            listenning();
        }
    }
}
```



# Lớp: ServerControl (2)

---

```
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());
    }
}

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



# Lớp: ServerControl (3)

---

```
private void listenning(){
    try {
        clientSocket = myServer.accept();
        ObjectInputStream ois = new
            ObjectInputStream(clientSocket.getInputStream());
        ObjectOutputStream oos = new
            ObjectOutputStream(clientSocket.getOutputStream());

        Object o = ois.readObject();
        if(o instanceof User){
            User user = (User)o;
            if(checkUser(user)){
                oos.writeObject("ok");
            }
            else
                oos.writeObject("false");
        }
    }catch (Exception e) {
        view.showMessage(e.toString());
    }
}
```



# Lớp: ServerControl (4)

---

```
private boolean checkUser(User user) throws Exception {
    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) {
        throw e;
    }
    return false;
}
```

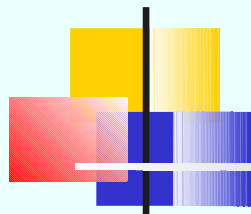


# Lớp: ServerRun

---

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

Lưu ý: chạy serverRun trước rồi chạy clientRun sau!

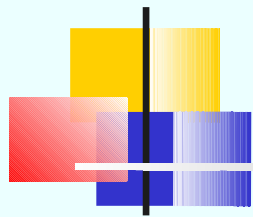


# Bài tập (1)

---

Cài đặt đúng mô hình MVC cải tiến cho bài toán quản lí người dùng theo TCP/IP:

- 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 cải tiến cho bài toán quản lí người dùng theo TCP/IP:

- 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?

---