HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG


BÀI GIẢNG MÔN

# Lập trình mạng

Giảng viên: TS. Nguyễn Trọng Khánh

Điện thoại/E-mail: khanhnt@ptit.edu.vn

Bộ môn: CNPM- Khoa CNTT1

Học kỳ/Năm biên soạn: 2021

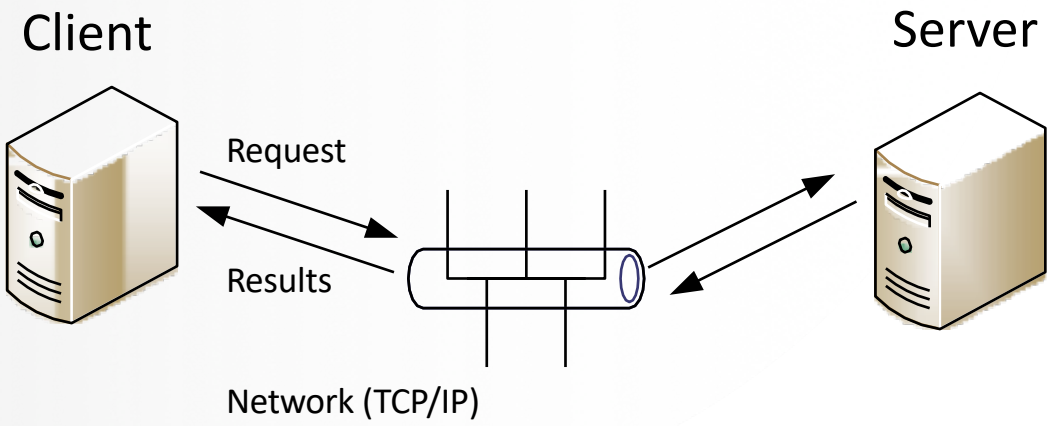
# Socket - TCP

# Nội dung

- Socket và liên lạc qua socket
- Tạo Server và client socket trong java
- Demo
- Bài tập

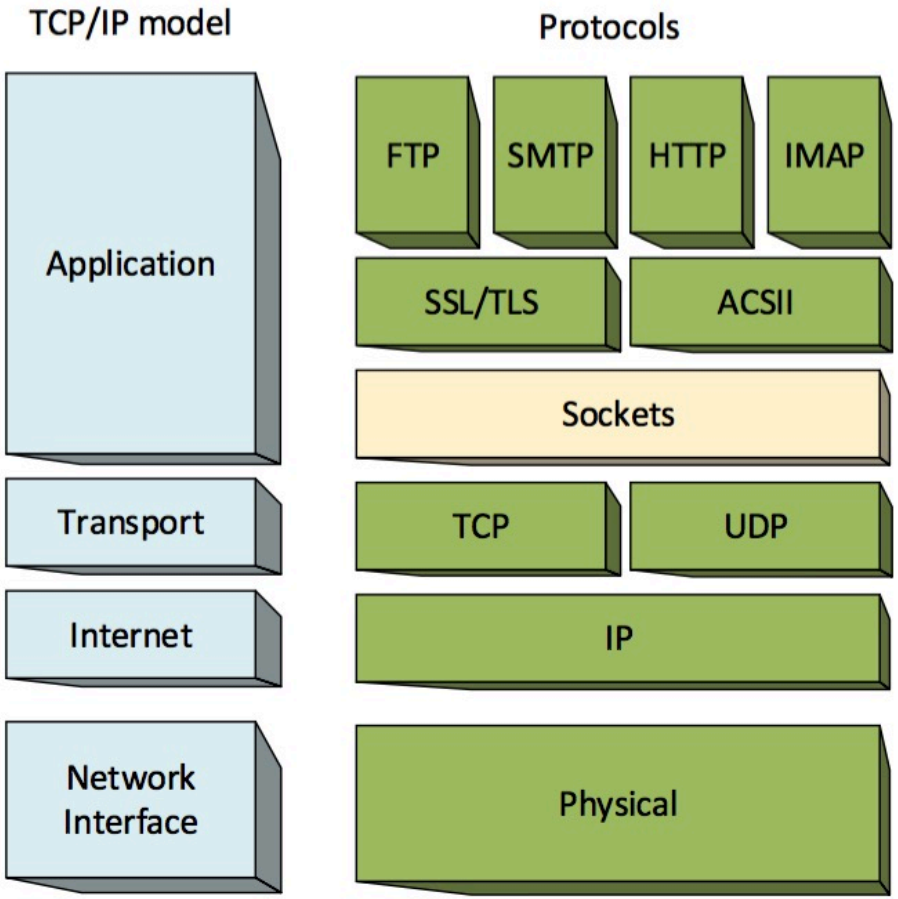
# Client/Server

- Mức độ liên lạc mạng đơn giản nhất
- Bao gồm: server, client, và môi trường kết nối
  - Client: máy tính chạy chương trình tạo yêu cầu dịch vụ
  - Server: máy tính chạy chương trình trả lời yêu cầu dịch vụ



# TCP/IP Application layer: Sockets

- Tầng thấp nhất nằm tiếp giáp tầng Giao vận
- Thiết lập một endpoint cho đường kết nối 2 chiều
- Cung cấp giao tiếp → lập trình liên lac mạng.
- Tương tự vào ra file, socket được xử lý tương tự file.
- Độc lập ngôn ngữ lập trình



# Socket

- Một socket bao gồm
  - Địa chỉ cục bộ của socket: Địa chỉ IP local và số cổng dịch vụ
  - Địa chỉ từ xa của socket: dành cho các socket thiết lập TCP
  - Giao thức: Giao thức tầng giao vận, ví dụ TCP hoặc UDP.
- Địa chỉ socket: kết hợp giữa địa chỉ IP và số cổng dịch vụ.
- API của socket: thường được cung cấp bởi hệ điều hành.

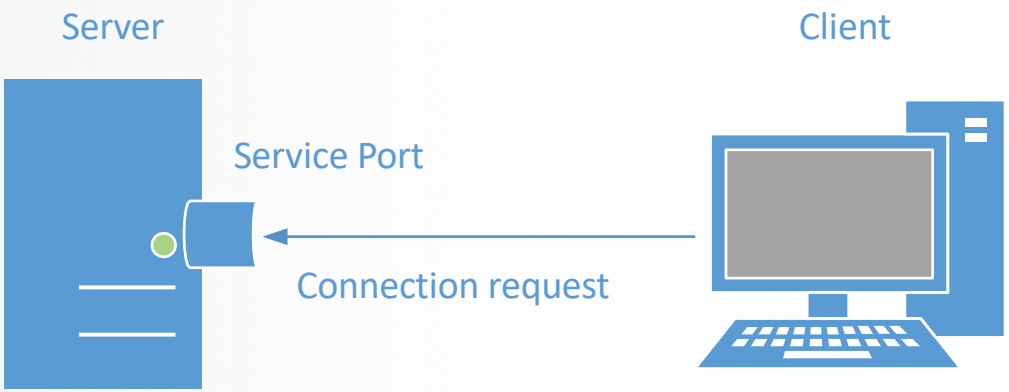
# Cổng dịch vụ

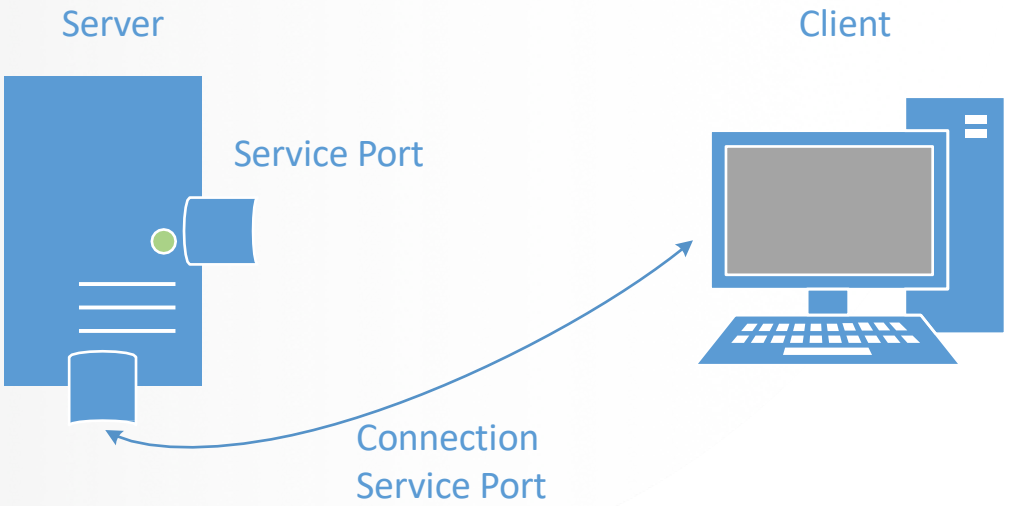
- Có nhiều dịch vụ, liên lạc với nhiều máy tính khác nhau tại mỗi thời điểm
- Cổng → phân biện các dịch vụ → mỗi dịch vụ có một cổng riêng
  - Số nguyên 16 bit
  - Một số cổng nổi tiếng:
    - FTP: 21/TCP
    - HTTP: 80/TCP,UDP
    - IMAP: 143/TCP

# Thiết lập kết nối Socket

- Step 1: Tạo server, server lắng nghe socket (với cổng cụ thể) để chờ yêu cầu kết nối từ client
- Step 2: Server chấp nhận kết nối
- Step 3: Server lấy về socket mới gắn với cổng khác (sinh ngẫu nhiên) để kết nối tới client
- Quay lại step 1
- Hỏi: cổng dịch vụ cho yêu cầu kết nối socket TCP khác với cổng để duy trình kết nối. Tại sao ?

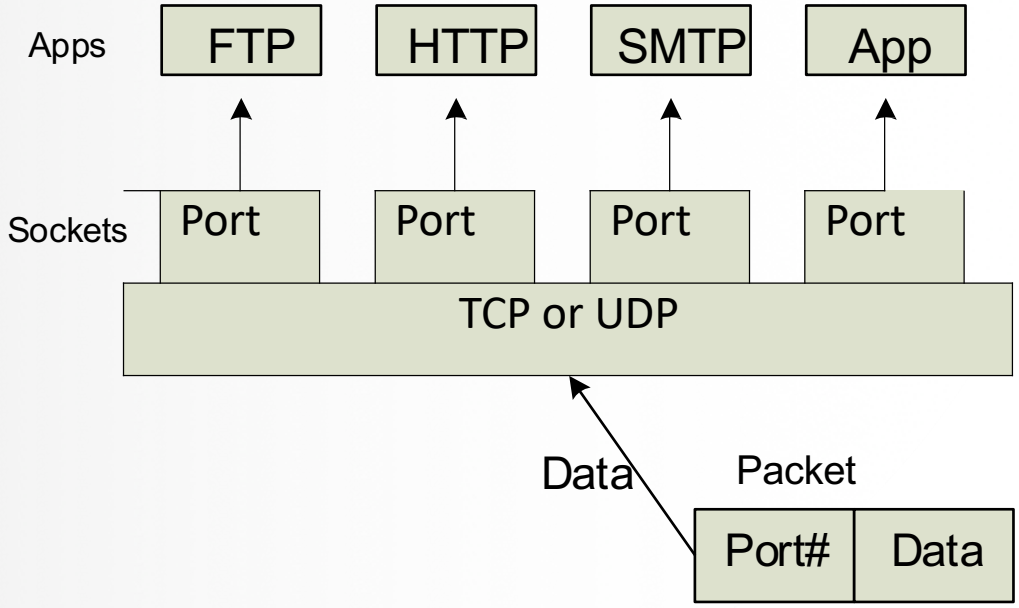
# Thiết lập kết nối Socket



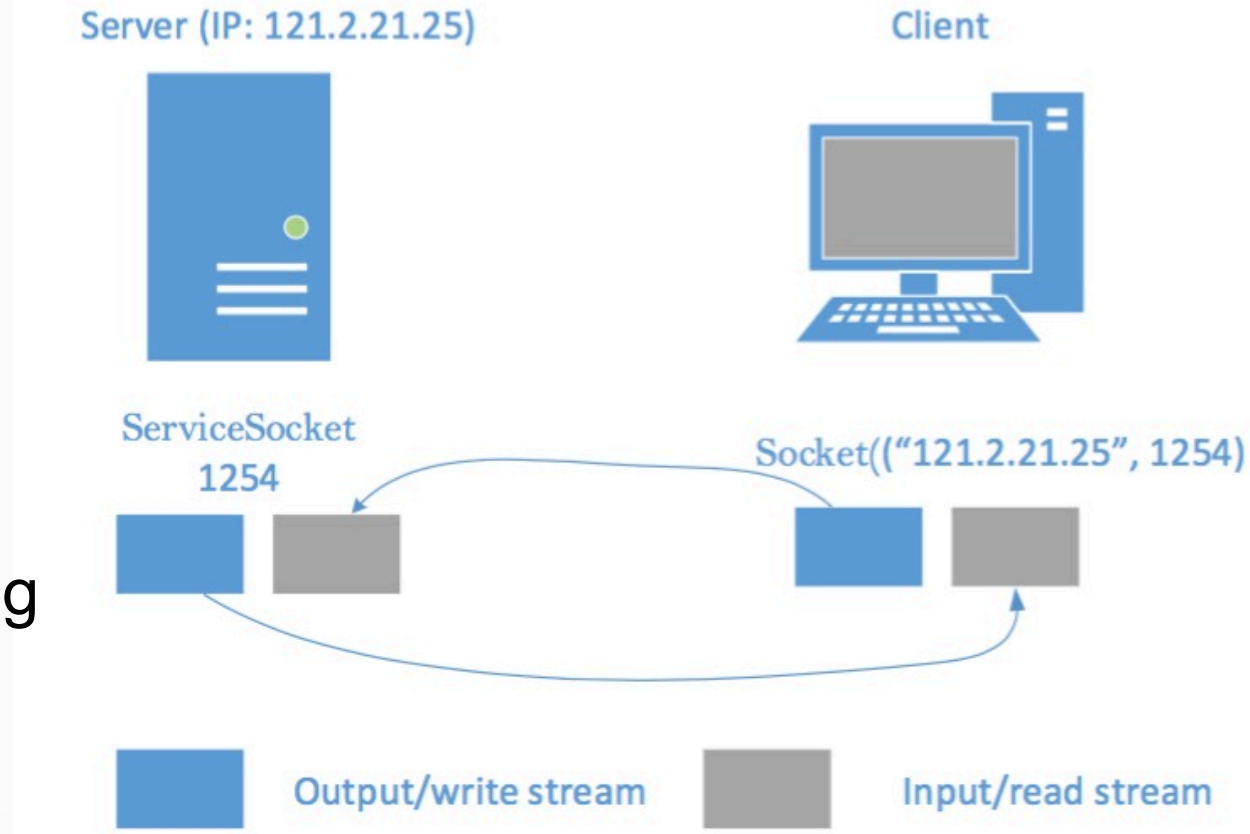
# Kết nối socket

- Client: kết nối tới server thông qua cổng
- Kết nối được xác định thông qua số cổng và số socket
- Cả giao thức TCP và UDP đề sử dụng cổng để chuyển dữ liệu tới các tiến trình tương ứng.



# Lớp Java.net

- Lớp thư viện cho phát triển ứng dụng mạng.
- 2 lớp quan trọng
  - Server: ***ServerSocket***
  - Client: ***Socket***
- Sử dụng các luồng vào ra để trao đổi dữ liệu



# Tạo chương trình với Java (1)

Các bước để tạo server:

- Step 1: Mở Socket server :

```
ServerSocket server = new ServerSocket(PORT);
```

- Step 2: Chờ yêu cầu kết nối từ client :

```
Socket client = server.accept();
```

- Step 3: Tạo luồng để liên lạc với client

```
DataInputStream is =
new DataInputStream(client.getInputStream());
DataOutputStream os =
new DataOutputStream(client.getOutputStream());
```

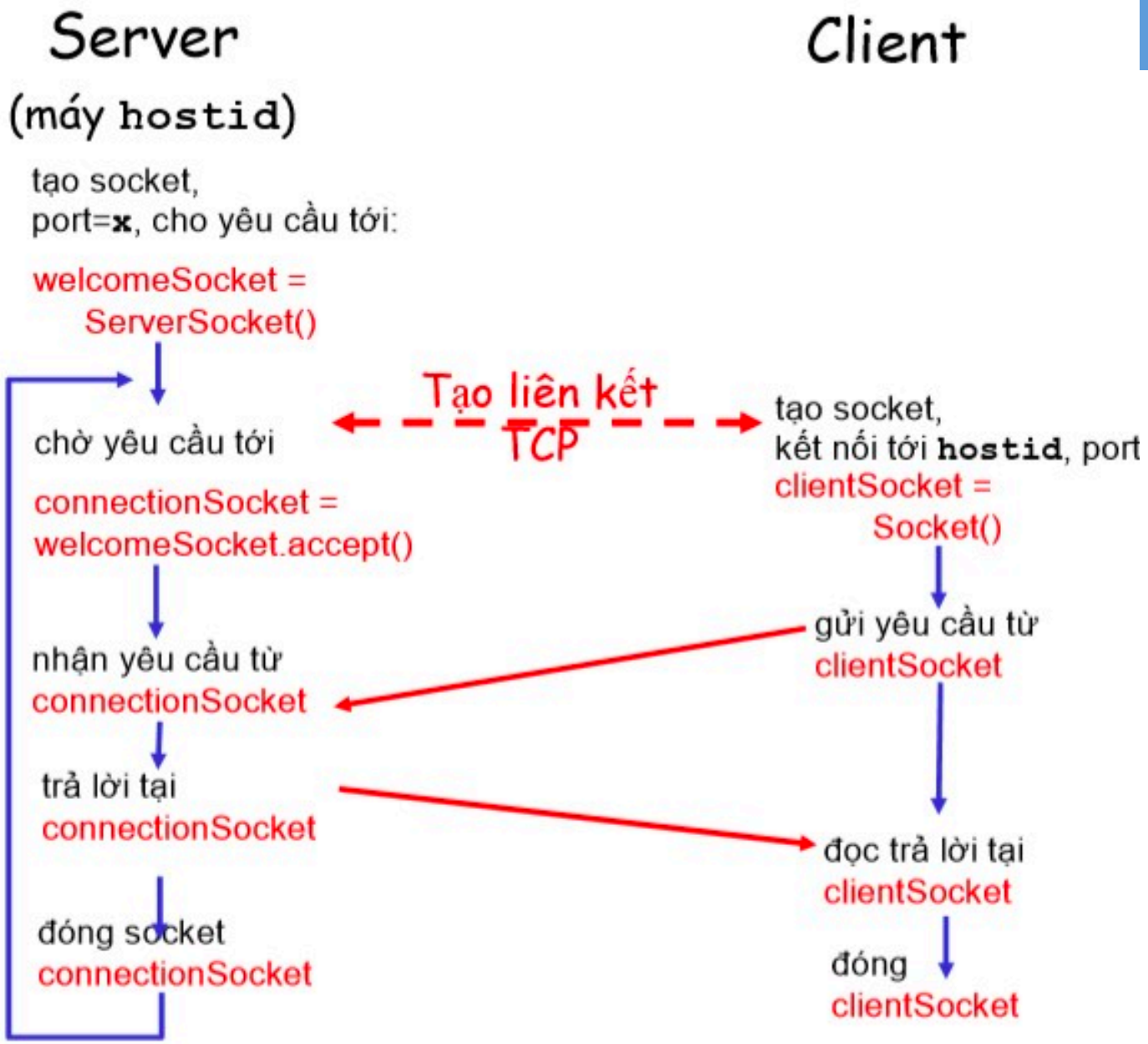
# Tạo chương trình với Java (2)

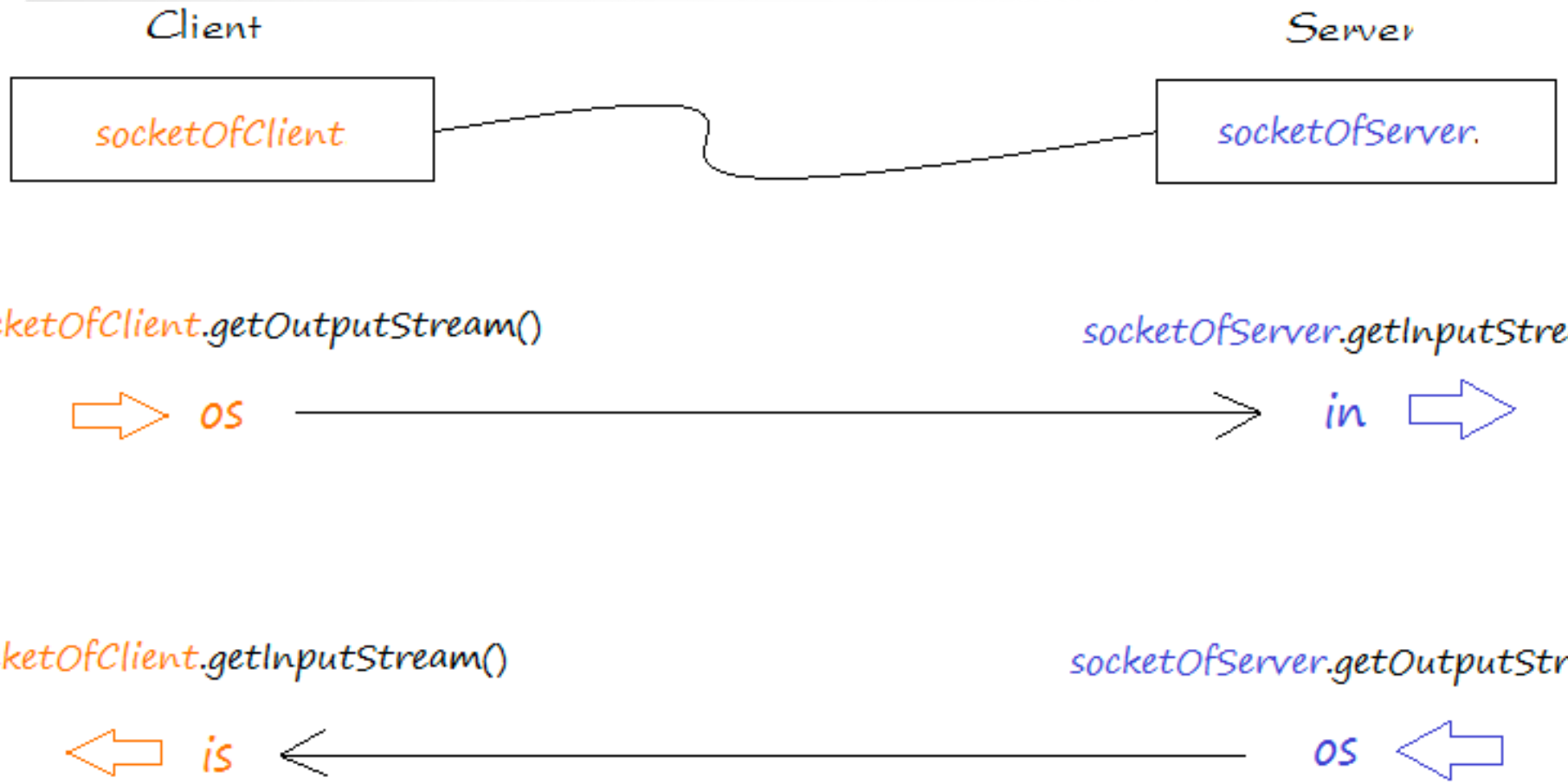
- Step 4: Trao đổi với client
  - Nhận dữ liệu : `String line = is.readLine();`
  - Gửi dữ liệu : `os.writeBytes("Hello!");`
- Step 5: Đóng kết nối `is.close();` `os.close();`
- Step 6: Đóng socket: `client.close();`

# Tạo chương trình với Java (3)

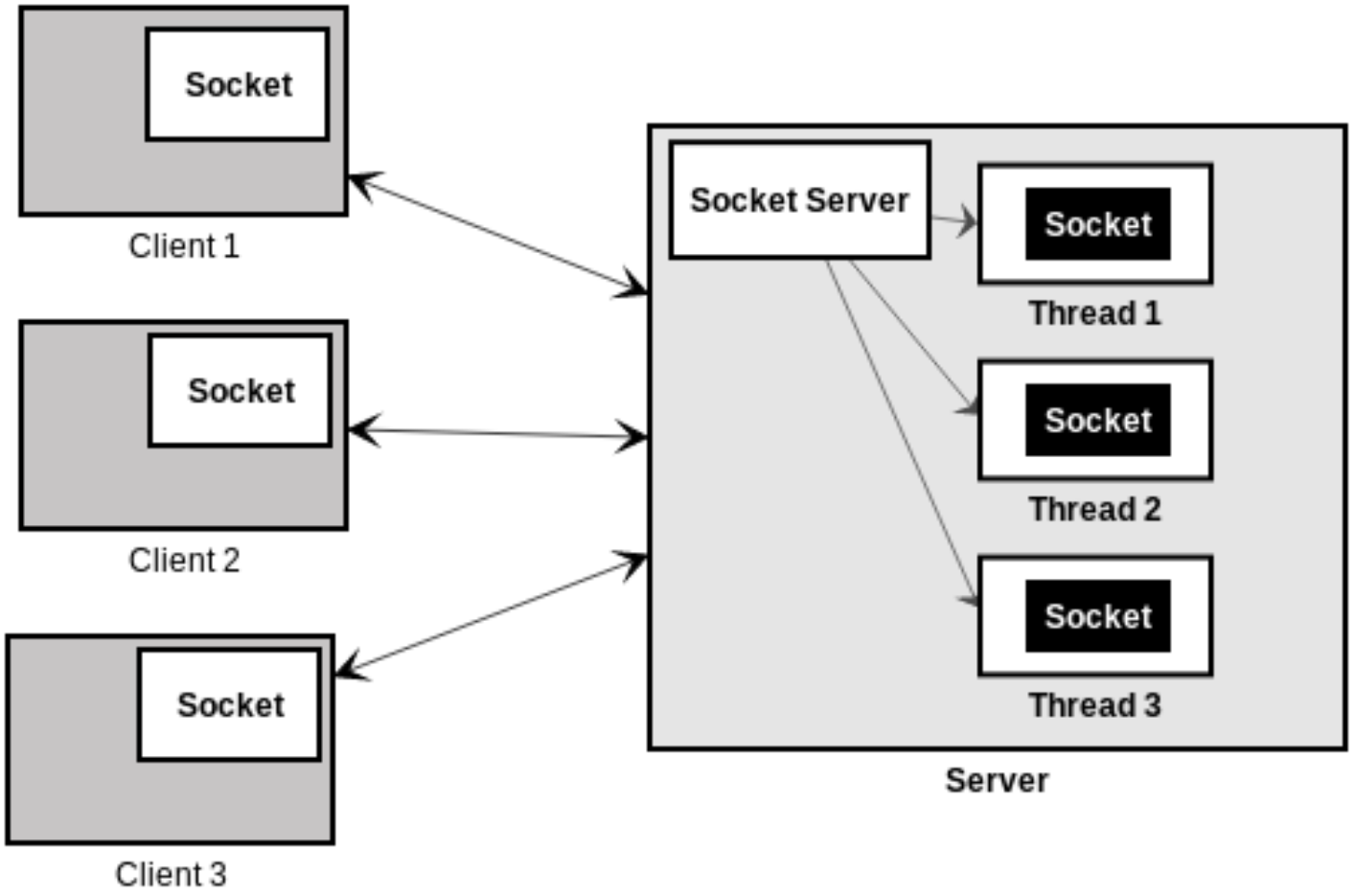
## Các bước để tạo client:

- Step 1:  Mở socket
- Step 2: Mở một luồng vào và ra tới socket
- Step 3: Đọc từ và ghi ra luồng theo giao thức phù hợp
- Step 4:  Đóng luồng.
- Step 5:  Đóng socket


Server
(máy hostid)
Client
tạo socket,
port=x, cho yêu cầu tới:
welcomeSocket =
ServerSocket()
chờ yêu cầu tới
Tạo liên kết
TCP
tạo socket,
kết nối tới hostid, port
clientSocket =
Socket()
connectionSocket =
welcomeSocket.accept()
gửi yêu cầu từ
clientSocket
nhận yêu cầu từ
connectionSocket
trả lời tại
connectionSocket
đọc trả lời tại
clientSocket
đóng socket
connectionSocket
đóng
clientSocket


Client
Server
socketOfClient
socketOfServer.
socketOfClient.getOutputStream()
socketOfServer.getInputStream()
os
in
socketOfClient.getInputStream()
socketOfServer.getOutputStream()
is
os

# Đa luồng


Socket
Client 1
Socket
Client 2
Socket
Client 3
Socket Server
Socket
Thread 1
Socket
Thread 2
Socket
Thread 3
Server

# Đa luồng

```
1  ServerSocket serverSocket = new ServerSocket(PORT_NUMBER);
2
3  while (true) {
4      Socket client = serverSocket.accept();
5      try {
6          BufferedReader in = new BufferedReader(
7              new InputStreamReader(client.getInputStream())
8          );
9          OutputStream out = client.getOutputStream();
10         in.lines().forEach(line -> {
11             try {
12                 out.write(line.getBytes());
13             } catch (IOException e) {
14                 e.printStackTrace();
15             }
16         });
17         client.close();
18     } catch (IOException e) {
19         e.printStackTrace();
20     }
21 }
22
```

# Một số lớp quan trọng

- InetAddress
  - Biểu diễn địa chỉ IP
  - Phương thức getLocalHost(): trả về địa chỉ IP
- Protocol

# Demo

# Bài tập

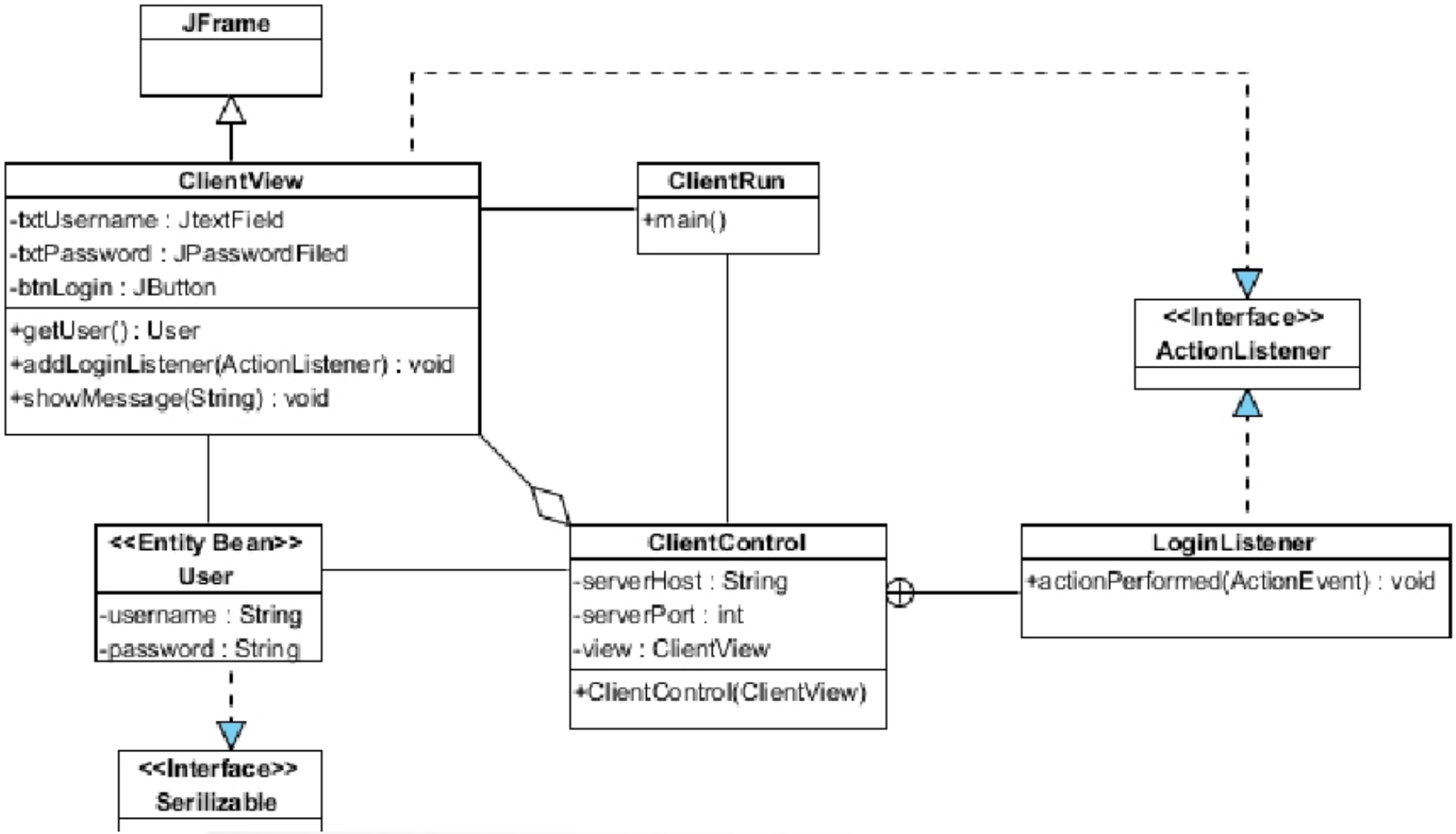
- Cài đặt theo mô hình giao thức TCI/IP cho bài toán:
  - Client yêu cầu người dùng nhập từ bàn phím hai số nguyên a và b
  - server nhận và tính tổng a và b, sau đó trả về kết quả cho client
  - Client nhận lại kết quả tổng và show ra màn hình cho người dùng
  - Yêu cầu gửi dữ liệu theo kiểu đối tượng

# Ví dụ:
# Login từ xa dùng TCP/IP

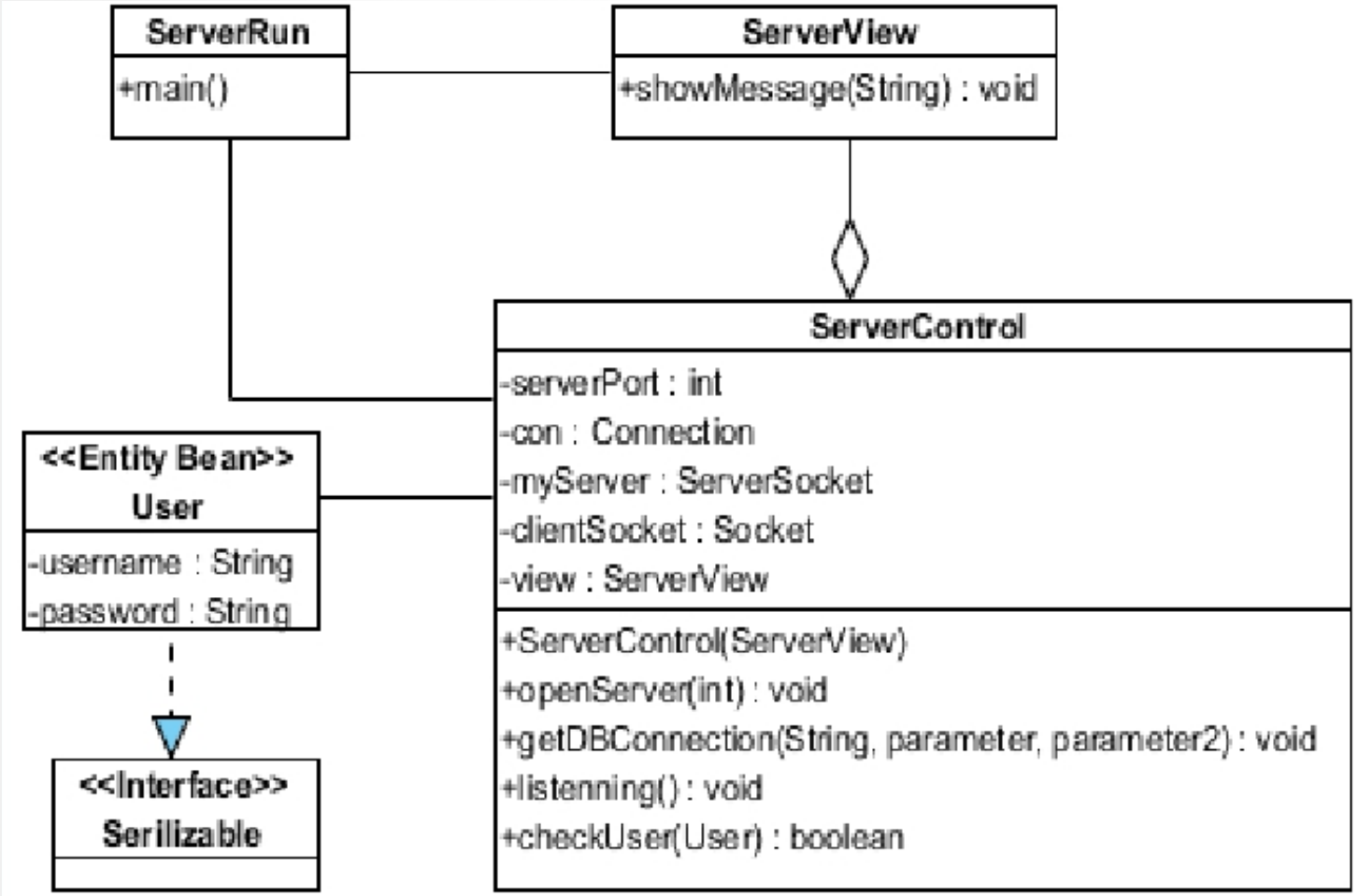
# Bài toán: Login dùng TCP/IP

- 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 lí
- Kết quả đăng nhập được trả từ server về client và
- client thông báo lại cho người dùng

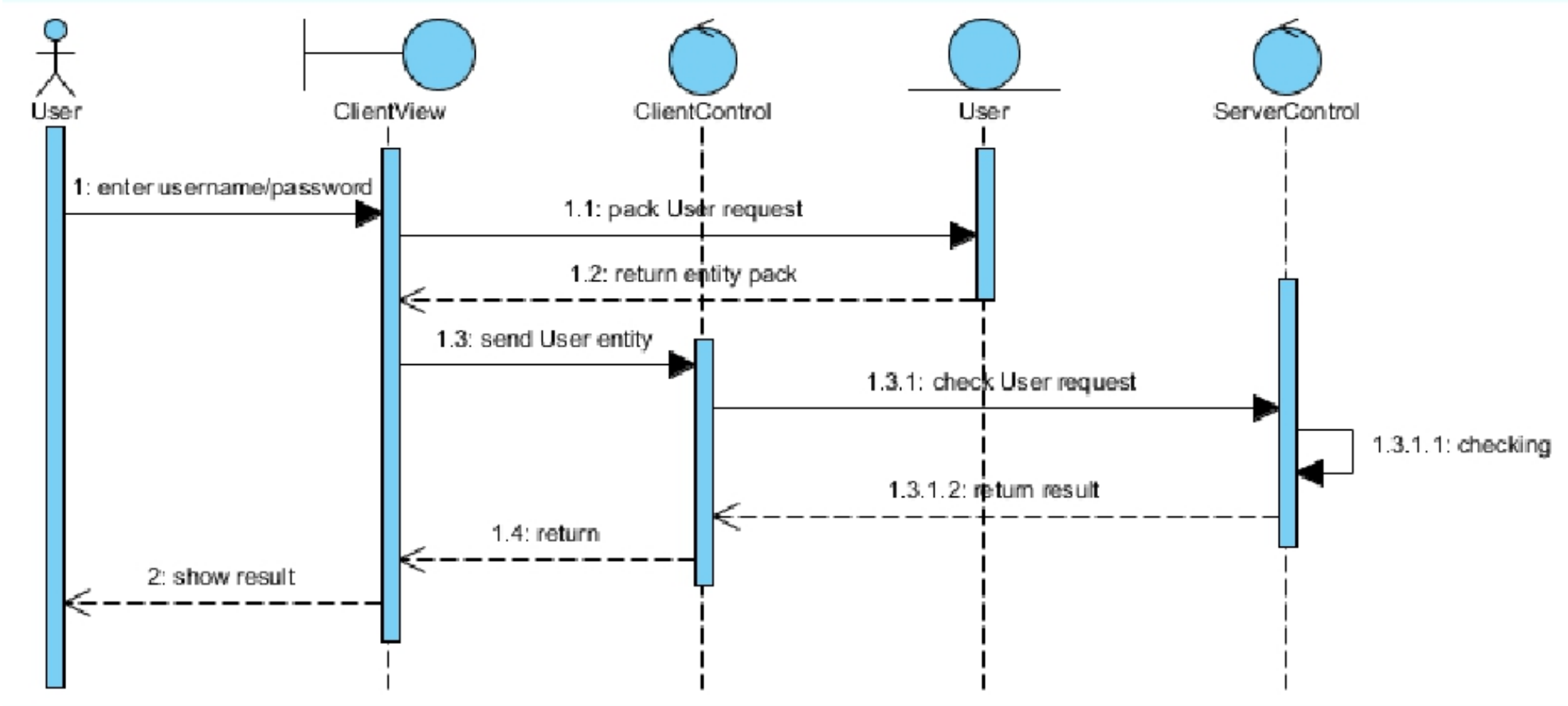
# Sơ đồ lớp phía client



# 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("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.showMessage("Login succesfully!");
            else view.showMessage("Invalid username and/or
password!");
            }
            mySocket.close();
            } catch (Exception ex) {
                view.showMessage(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!**

# Questions?