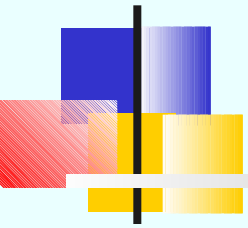


Lập trình mạng

Lập trình Socket với TCP/IP

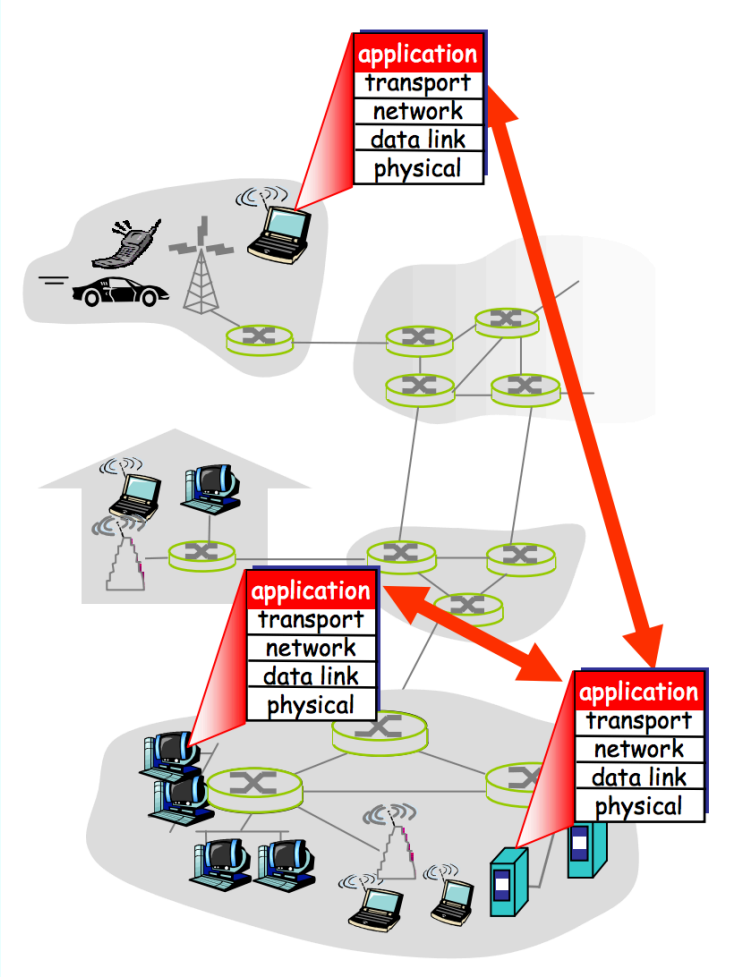
Tổng quan

Viết chương trình

– Chạy trên các hệ thống đầu cuối

– Truyền thông qua mạng

– Ví dụ web server giao tiếp với browser



Viết chương trình trên các thiết bị mạng ngoài phạm vi này

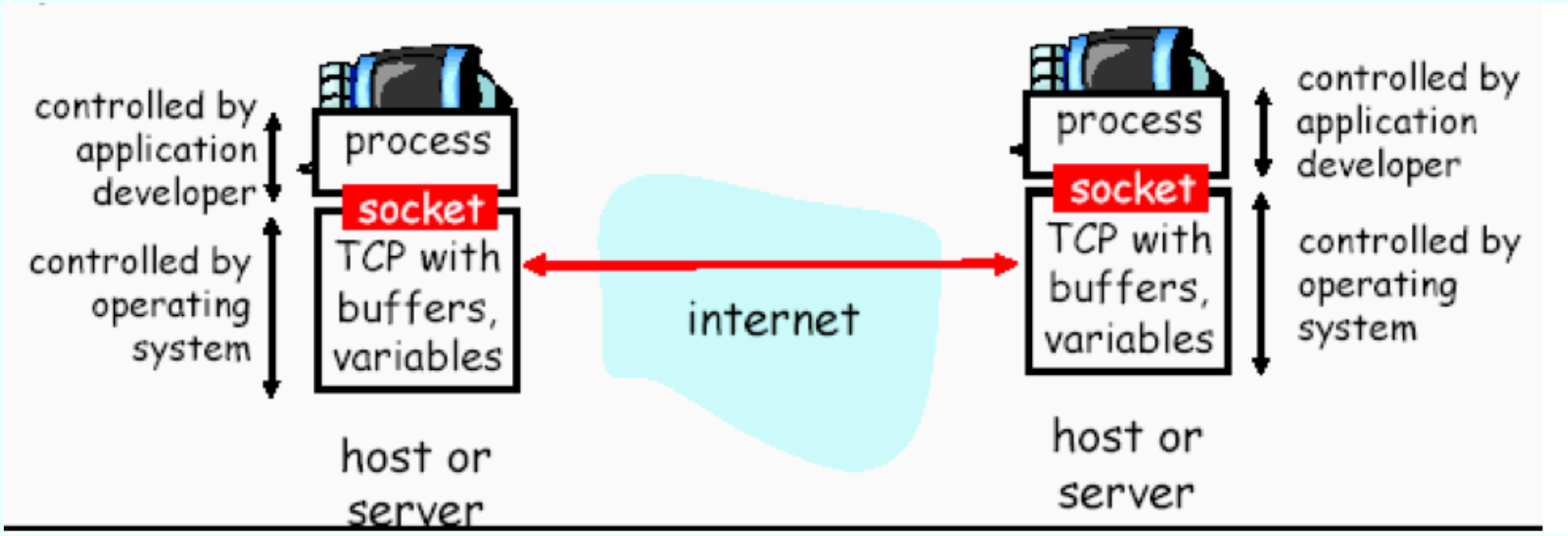
LẬP TRÌNH SOCKET

Mục tiêu: biết cách xây dựng một ứng dụng client/server giao tiếp bằng socket

- **Socket API**
  - Được giới thiệu ở BSD4.1 UNIX, 1981
  - Được ứng dụng khởi tạo, sử dụng và hủy bỏ.
  - Dùng cơ chế client/server
  - Cung cấp hai dịch vụ chuyển dữ liệu thông qua socket API:
    - unreliable datagram
    - reliable, byte stream-oriented

KHÁI NIỆM VỀ SOCKET

- Socket: “cửa” nằm giữa process ứng dụng và end-end-transport protocol (UDP hoặc TCP)
- TCP Service: dịch vụ truyền tin cậy chuỗi byte giữa 2 process



Lập trình Socket với TCP

Client phải liên lạc với server

- Trước hết quá trình trên server phải chạy
- server phải tạo socket để đón tiếp client

Client liên lạc server bằng:

- Tạo TCP socket
- Chỉ ra IP address, port number của quá trình trên server
- Khi client tạo socket: client TCP thiết lập kết nối đến server TCP

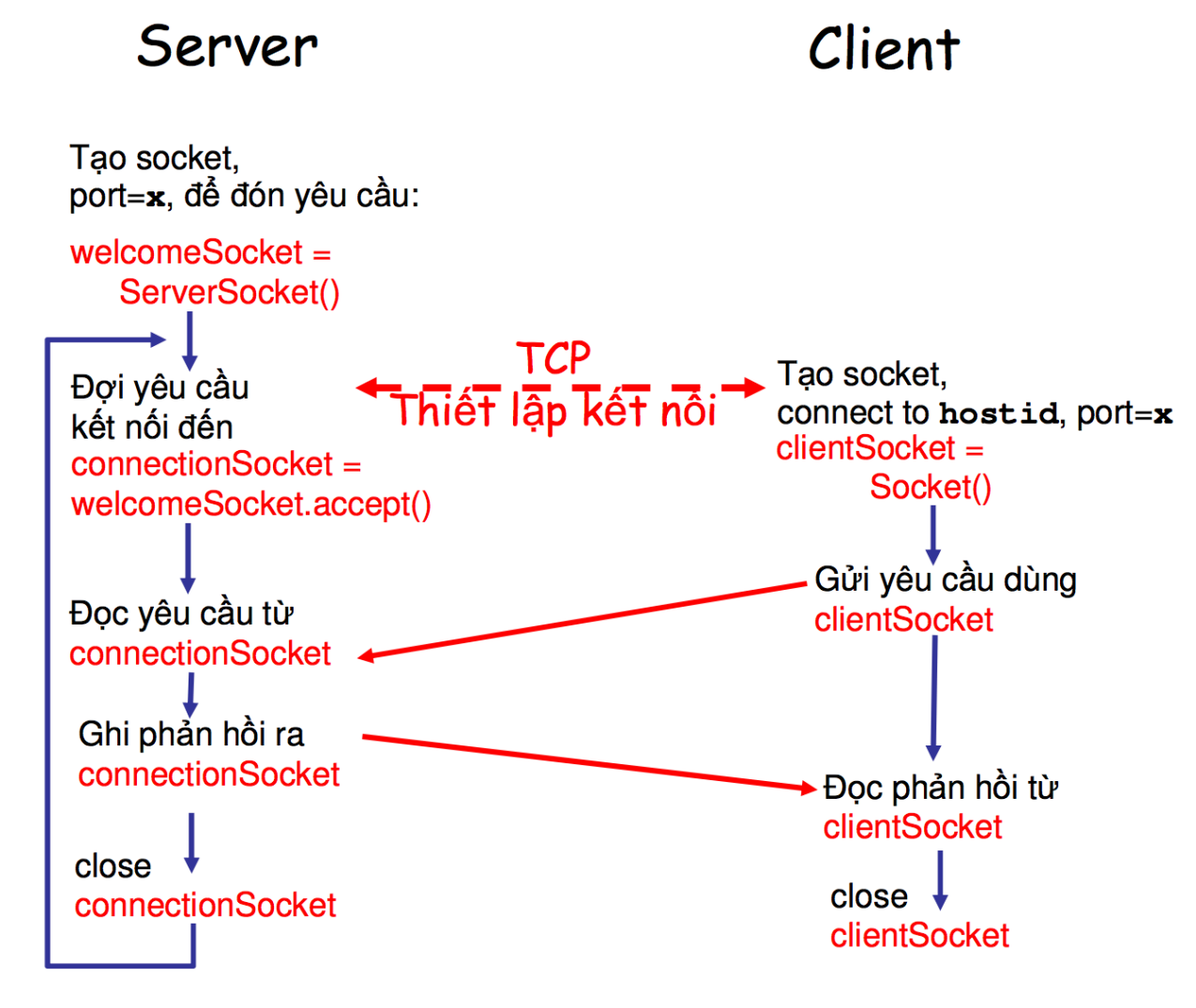
Khi được liên hệ bởi client, server TCP tạo socket mới để quá trình server giao tiếp với client

– Cho phép server giao tiếp v ới nhiều quá trình client

– Các chỉ số port được dùng để phân biệt các client

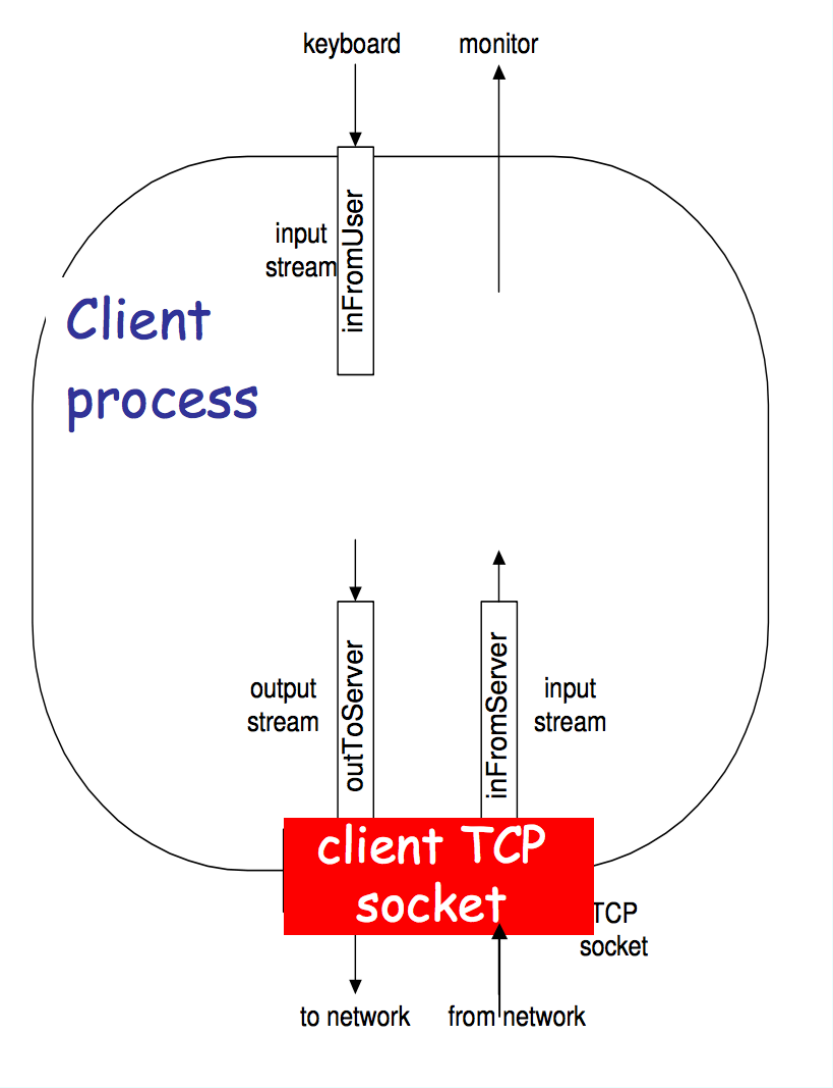
Từ góc độ ứng dụng:
TCP cung cấp dịch vụ truyền tải có trật tự và bảo đảm giữa client và server

Tương tác giữa client socket và server socket qua TCP



Thuật ngữ Stream

- **Stream** là một tuần tự các ký tự đi vào hay đi ra một quá trình.
- Một **input stream** được kết với một nguồn nhập của quá trình, ví dụ bàn phím hay socket.
- Một **output stream** được kết với ột đích đến của quá trình, ví dụ màn hình hay socket.



Thiết Kế Giải Thuật Client/Server

- **Thiết kế giải thuật cho client**

  - Giải thuật cho chương trình client dùng TCP
    - Xác định địa chỉ server
    - Tạo socket.
    - Kết nối đến server.
    - Gởi/nhận dữ liệu theo giao thức lớp ứng dụng đã thiết kế.
    - Đóng kết nối.

Thiết Kế Giải Thuật Client/Server

- **Thiết kế giải thuật cho Server**
  - Chương trình server có hai loại:
    - Lặp(iterative)
    - Đồng thời (concurrent).

Thiết Kế Giải Thuật Client/Server

- Giải thuật cho chương trình server iterative, connection-oriented:
  - Tạo Socket và đăng ký địa chỉ socket với hệ thống
  - Đặt socket ở trạng thái lắng nghe, chờ và sẵn sàng cho việc kết nối từ client
  - Chấp nhập kết nối từ client, gởi/nhận dữ liệu theo giao thức lớp ứng dụng đã thiết kế
  - Đóng kết nối sau khi hoàn thành, trở lại trạng thái lắng nghe và chờ kết nối mới

Thiết Kế Giải Thuật Client/Server

- Giải thuật cho chương trình server concurrent, connection-oriented:
  - Tạo Socket và đăng ký với hệ thống
  - Đặt socket ở chế đồ chờ, lắng nghe kết nối
  - Khi có request từ client, chấp nhật kết nối, tạo mới một process để xử lý. Quay lại trạng thái chờ, lắng nghe kết nối mới
  - Công việc của process mới:
    - Nhận thông tin của process cha chuyển đến, lấy thông tin socket
    - Xử lý và gởi thông tin về cho client theo giao thức lớp ứng dụng đã thiết kế
    - Đóng kết nối và kết thúc process con

Lập Trình Mạng Trên Java

- ***InetAddress* class**
  - Class mô tả về địa chỉ IP (Internet Protocol)
  - Các phương thức *getLocalHost*, *getByName*, hay *getAllByName* để tạo một *InetAddress* instance:
    - *public static InetAddess InetAddress.getByName(String hostname)*
    - *public static InetAddess [] InetAddress.getAllByName(String hostname)*
    - *public static InetAddess InetAddress.getLocalHost()*
  - Để lấy địa chỉ IP hay tên dùng các phương thức:
    - *getHostAddress()*
    - *getHostName()*

Lập Trình Mạng Trên Java

- **In địa chỉ IP của localhost**

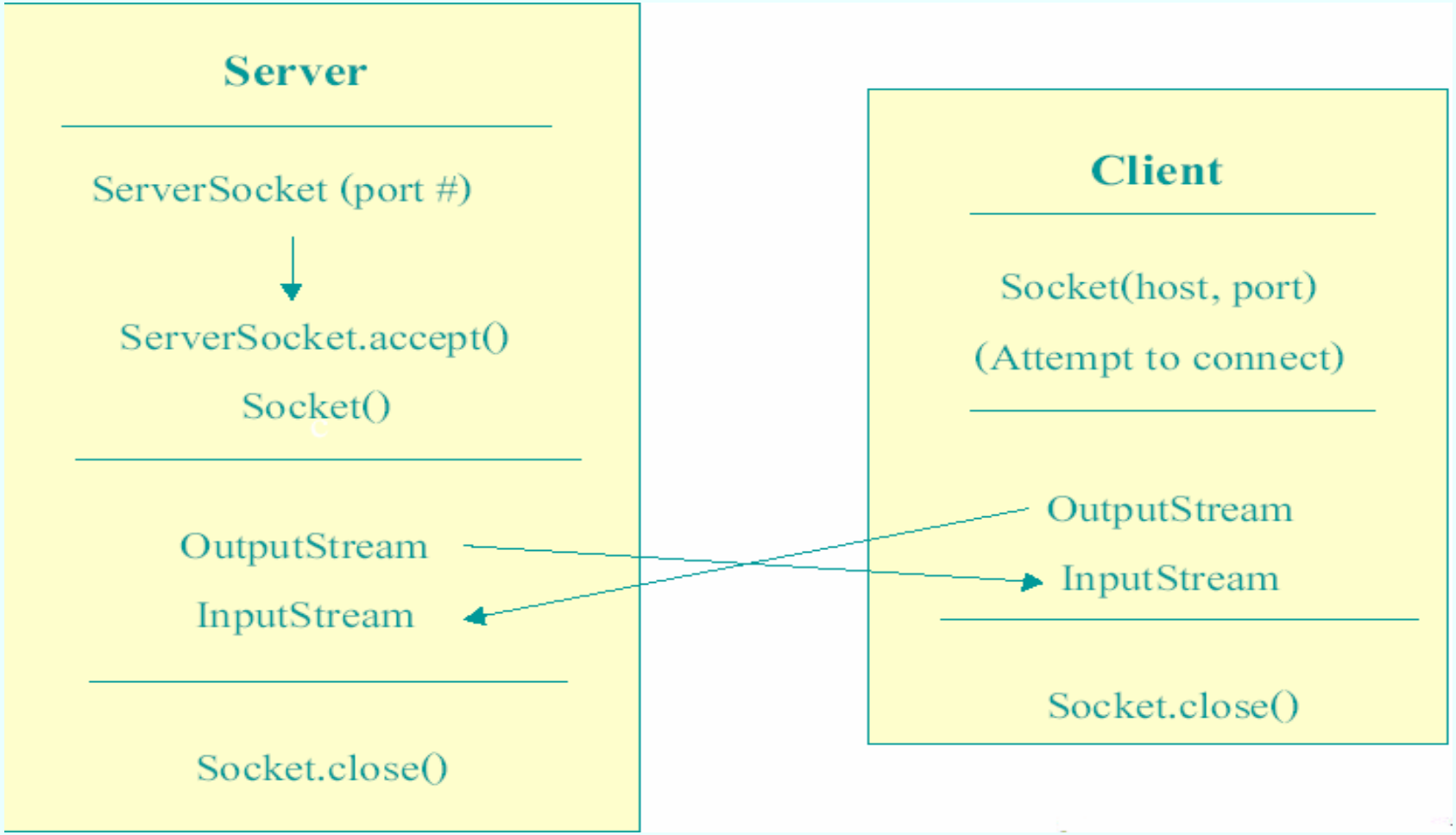
```
import java.net.*;
public class HostInfo {
  public static void main(String args[]) {
  HostInfo host = new HostInfo();  host.init();
  }
  public void init() {
  try {
          InetAddress myHost = InetAddress.getLocalHost();
          System.out.println(myHost.getHostAddress());
          System.out.println(myHost.getHostName());
     } catch (UnknownHostException ex) {
          System.err.println("Cannot find local host");
     }
  }
}
```

Lập Trình Mạng Trên Java

- **In địa chỉ IP của google.com**

```
import java.net.*;  class kku{
   public static void main (String args[]) {  try {
                InetAddress[] addresses =
                InetAddress.getAllByName("google.com  ");
                for (int i = 0; i < addresses.length;  i++) {
                        System.out.println(addresses[i]);
                }
        }
        catch (UnknownHostException  e) {
                System.out.println("Could not find  google.com ");
        }
   }
}
```

Lập Trình Mạng Trên Java



Lập Trình Mạng Trên Java

- ***Socket* class**
  - Class mô tả về *socket*
  - Tạo một *socket*
    - **Socket**(InetAddress address, int port)
    - **Socket**(String host, int port)
    - **Socket**(InetAddress address, int port, InetAddress, localAddr, int localPort)
    - **Socket**(String host, int port, InetAddress, localAddr, int localPort)
    - **Socket**()

Lập Trình Mạng Trên Java

- ***Socket*** **class (tiếp theo)**
  - **Lấy thông tin về một *socket***
    - InetAddress **getInetAddress**() : trả về địa chỉ mà socket kết nối đến.
    - int **getPort**() : trả về địa chỉ mà socket kết nối đến.
    - InetAddress **getLocalAddress**() : trả về địa chỉ cục bộ.
    - int **getLocalPort**() : trả về địa chỉ cục bộ.
  - **Sử dụng Streams**
    - public OutputStream **getOutputStream**() throws IOException Trả về một output stream cho việc viết các byte đến socket này.
    - public InputStream **getInputStream()** throws IOException Trả về một input stream cho việc đọc các byte từ socket này

Lập Trình Mạng Trên Java

- ***ServerSocket* class**
  - Class mô tả về *ServerSocket*
  - Tạo một *ServerSocket*
    - **ServerSocket**(int port) throws IOException
    - **ServerSocket**(int port, int backlog) throws IOException
    - **ServerSocket**(int port, int backlog, InetAddress bindAddr) throws IOException

Lập Trình Mạng Trên Java

- ***ServerSocket* class**
  - Các phương thức trong *ServerSocket*
    - Socket **accept**() throws IOException : Lắng nghe một kết nối đến socket này và chấp nhận nó.
    - void **close**() throws IOException : Đóng socket.
    - InetAddress **getInetAddress**() : trả về địa chỉ cục bộ của socket
    - int **getLocalPort**() : Trả về port mà server đang lắng nghe.
    - void **setSoTimeout**(int timeout) throws SocketException
    - Enable/disable SO_TIMEOUT với khai báo timeout (milliseconds)

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 lí và gửi trả về client đó

```
// Xu li du lieu nhan duoc va tra ve
while (true) {
    // đồng ý và tạo kết nối với
    client
    .....
    // 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);
}
```

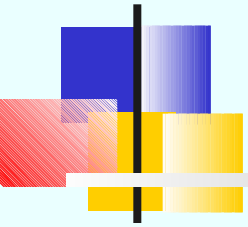
Bài tập (1)

- 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)

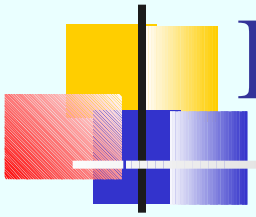
Cùng yêu cầu, nhưng cài đặt đúng mô hình MVC

- 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



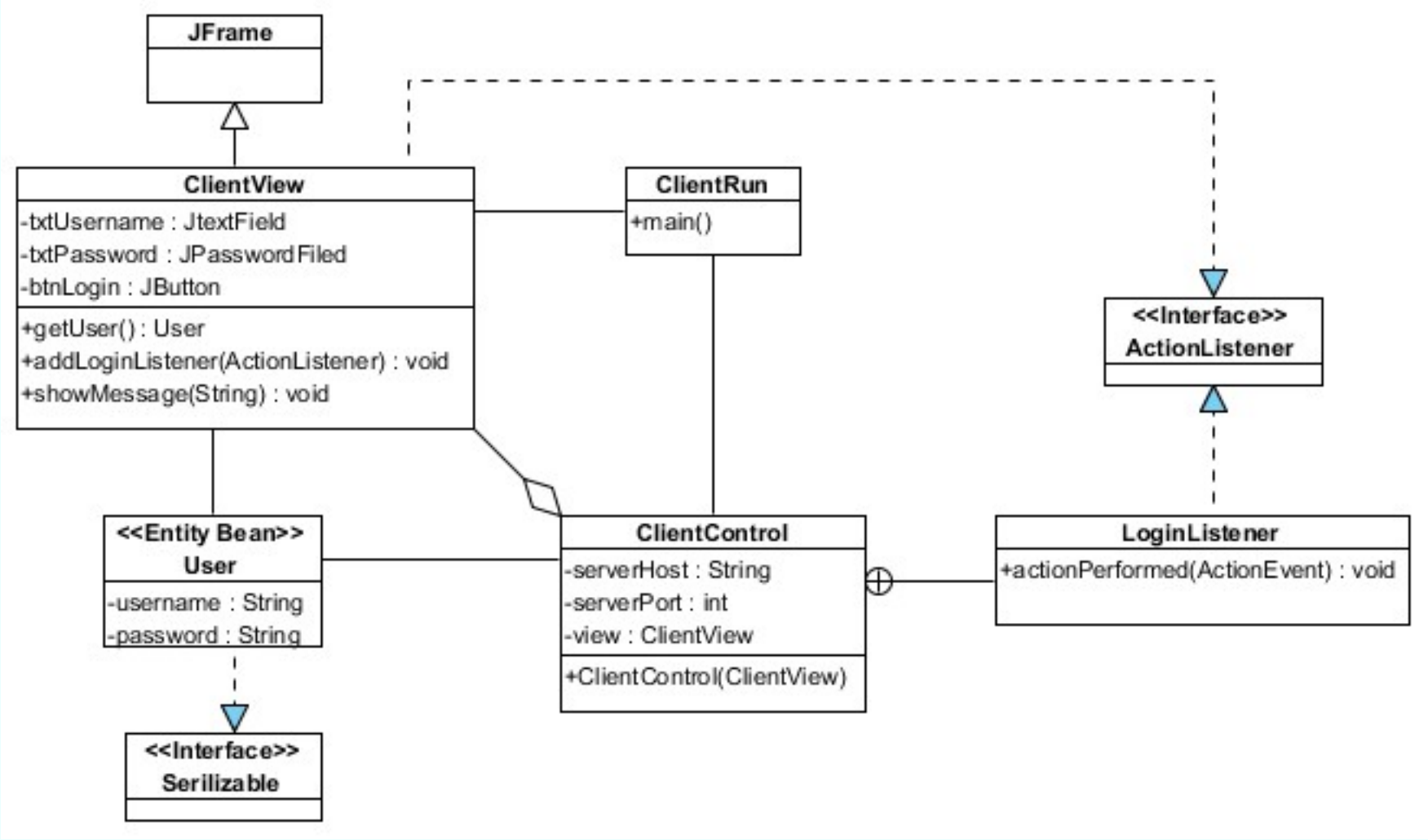
Ví dụ:
Login từ xa dùng TCP/IP

Bài toán: Login dùng TCP/IP

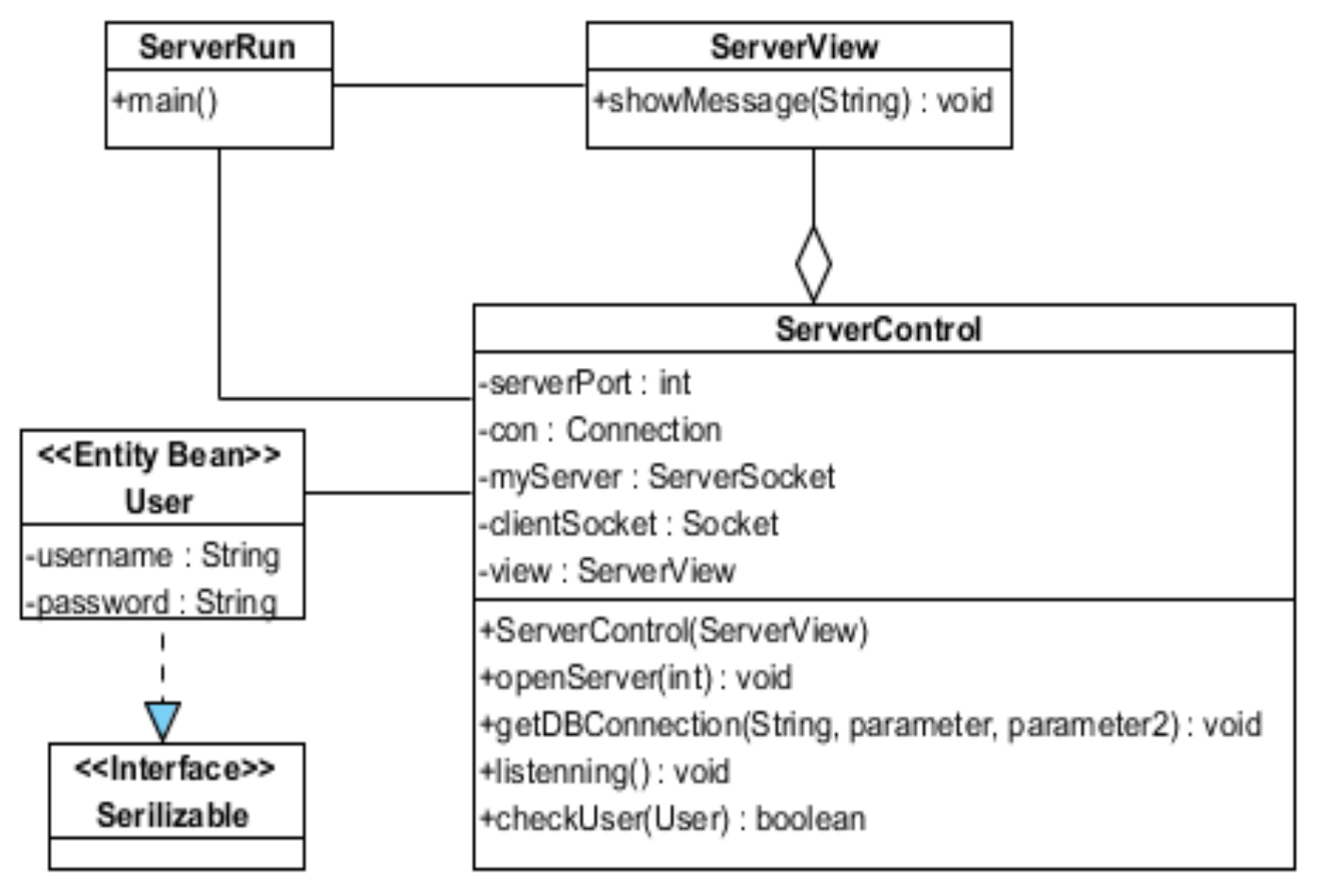


- Thông tin user được lưu trên server TCP
- Chương trình hiện cửa sổ đăng nhập GUI (username, password) ở phía client TCP
- Khi click vào nút login, client sẽ gửi thông tin đăng nhập lên server để xử lí
- Kết quả đăng nhập được trả từ server về client và client thông báo lại cho người dùng

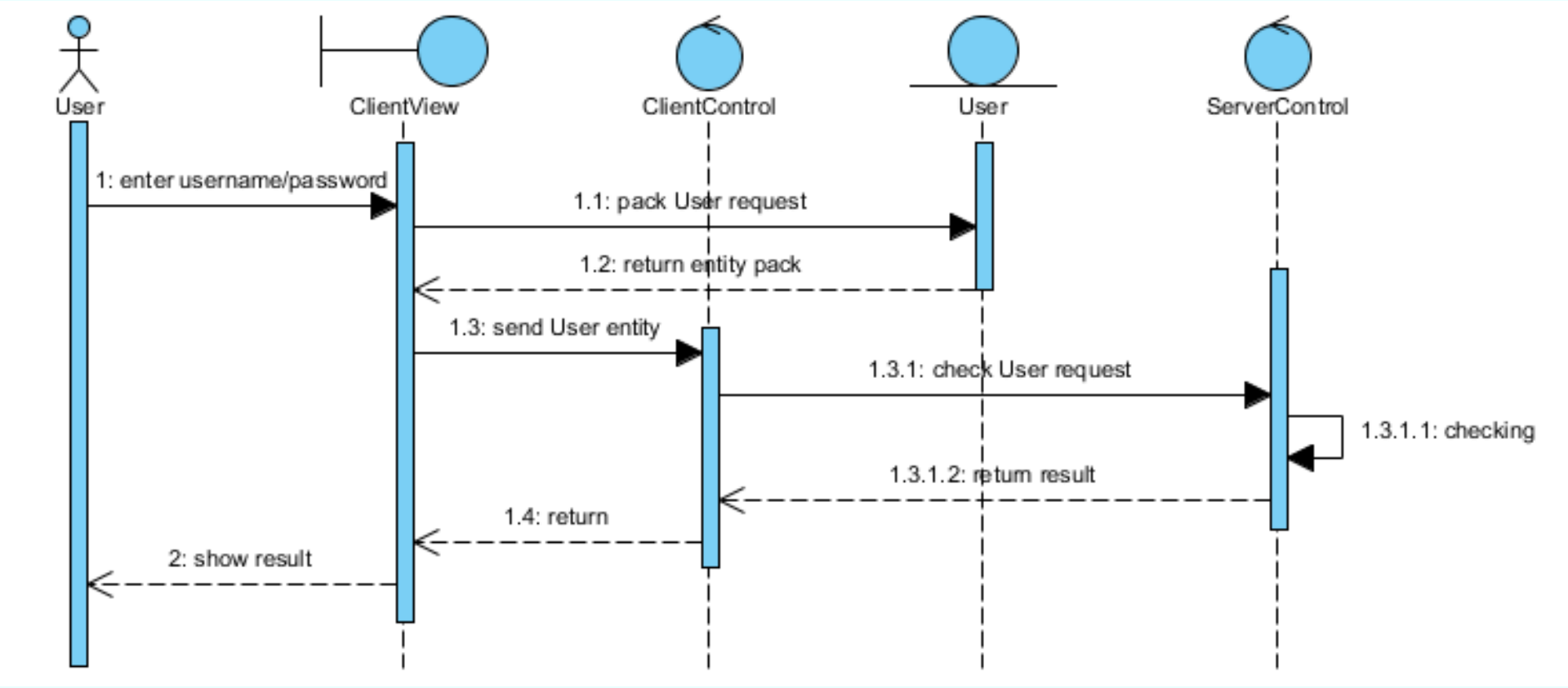
Sơ đồ lớp phía client



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());
    }
}
```

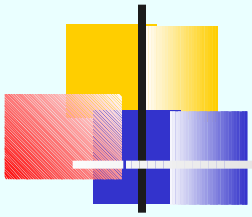
Lớp: ServerControl (4)

```
    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);
    }
}
```

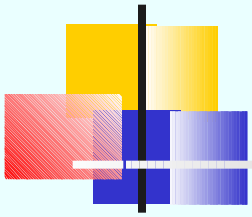
Lưu ý: chạy serverRun trước rồi chạy clientRun sau!



Bài tập (1)

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

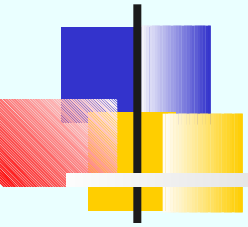
- Server chứa CSDL về người dùng, có bảng tbluser chứa các cột: id, username, password, address, birthday, sex, description
- Client có giao diện nhập thông tin đăng kí người dùng mới
- Sau khi nhập thông tin và click submit, client gửi thông tin đăng kí đến server
- Server kiểm tra xem có trùng username 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 (cũ hoặc mới) 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?