# BỘ CÔNG THƯƠNG

# TRƯỜNG ĐẠI HỌC KINH TẾ - KĨ THUẬT CÔNG NGHIỆP

# KHOA CÔNG NGHỆ THÔNG TIN

# 🙞🙞🙞🙞🕮🙞🙞🙞🙞

# Logo Description automatically generated with low confidence

**BÀI TẬP NHÓM**

**CÔNG NGHỆ JAVA**

**Giảng viên hướng dẫn : Vũ Văn Đốc**

**Thành viên tham gia : Nguyễn Đức Anh Lớp DHTI13A1ND**

**Trần Công Tuấn Anh Lớp DHTI13A1ND**

**Phạm Hoàng Hải Lớp DHTI13A1ND**

**Phạm Văn Hiếu Lớp DHTI13A1ND**

**Trần Huy Hoàng Lớp DHTI13A1ND**

**Nam Định 07/2022**

MỤC LỤC

[I. Một số lớp Java hỗ trợ lập trình với UDP Socket 3](#_Toc109080966)

[1.1. Lớp DatagramPacket 3](#_Toc109080967)

[Ví dụ 1 3](#_Toc109080968)

[1.2. Lớp DatagramSocket](#_Toc109080969)

[Ví dụ 2 4](#_Toc109080969)

[Ví dụ 3 5](#_Toc109080970)

[Ví dụ 4 7](#_Toc109080971)

[II. Kỹ thuật lập trình truyền thống với giao thức UDP 8](#_Toc109080972)

[Một số chương trình ví dụ lập trình giao thức UDP](#_Toc109080973)

[Ví dụ 5 9](#_Toc109080973)

[Ví dụ 6 11](#_Toc109080974)

[Lớp phía client User.java 14](#_Toc109080975)

[Client View. java 14](#_Toc109080976)

[ClientControl.java 16](#_Toc109080977)

[ClientRun.java 19](#_Toc109080978)

[Các lớp phía server 19](#_Toc109080979)

[ServerControl.java 19](#_Toc109080980)

[ServerRun.java 22](#_Toc109080981)

**Lập trình ứng dụng mạng với UDP Socket**

I. Một số lớp Java hỗ trợ lập trình với UDP Socket

1.1. Lớp DatagramPacket

Lớp này cho phép tạo gói tin truyền thông với giao thức UDP, kế thừa trực tiếp từ lớp Object.  
 public final class DatagramPacket extends Object  
Gói tin là đối tượng của lớp này chứa 4 thành phần quan trọng: Địa chỉ, dữ liệu truyền  
thật sự, kích thước của gói tin và số hiệu cổng chứa trong gói tin.  
 Các hàm tạo- Hàm tạo tạo gói tin nhận từ mạng:  
 public DatagramPacket(byte[] inBuffer, int length)  
Tham số:  
•inBuffer: Bộ đệm nhập, chứa dữ liệu của gói tin nhận  
•length: kích cỡ của dữ liệu của gói tin nhận, nó thường được xác định bằng  
lệnh: length= buffer.length.  
Tạo gói tin nhận:  
 byte[] inBuff=new byte[512];//bộ đệm nhập DatagramPacket inData=new  
 DatagramPacket (inBuf, inBuff.length);  
- Hàm tạo tạo gói tin gửi:  
 public DatagramPacket(byte[] outBuffer, int length, InetAddress destination, int port)  
Tham số:  
•outBuffer: Bộ đệm xuất chưa dữ liệu của gói tin gửi  
•length: Kích cỡ dữ liệu của gói tin gửi tính theo số byte và thường bằng  
outBuffer.length.  
•destination: Địa chỉ nơi nhận gói tin.  
•port: Số hiệu cổng đích, nơi nhận gói tin.  
Ví dụ 1 :

|  |
| --- |
| String s=" Hello World!"; //Bộ đệm xuất và gán dữ liệu cho bộ đệm xuất byte[] outBuff=s.getBytes(); //Địa chỉ đích InetAddressaddrDest=InetAddress.getByName("localhost"); //Số cổng đích int portDest=3456; //Tạo gói tin gửi DatagramPacket outData=new DatagramPacket(outBuff, outBuff.length, addrDest, portDest); |

Các phương thức  
•public InetAddress getAddress(): Phương thức này trả về đối tượng InetAddress  
của máy trạm từ xa chứa trong gói tin nhận.  
•public int getPort(): Trả về số hiệu cổng của máy trạm từ xa chứa trong gói tin.  
•public byte[] getData() : Trả về dữ liệu chứa trong gói tin dưới dạng mảng byte.  
•public int getLength(): Trả về kích cỡ của dữ liệu chưa trong gói tin tính theo số  
byte.  
Tương ứng với 4 phương thức getXXXX..(), lớp DatagramPacket có 4 phương thức  
setXXXX..() để thiết lập 4 tham số cho gói tin gửi.

1.2. Lớp DatagramSocket  
Lớp DatagramSocket cho phép tạo ra đối tượng socket truyền thông với giao thức  
UDP. Socket này cho phép gửi/nhận gói tin DatagramPacket.  
Lớp này được khai báo kế thừa từ lớp Object.  
public class DatagramSocket extends Object  
Các hàm tạo  
•public DatagramSocket() throws SocketException:  
Hàm tạo này cho phép tạo ra socket với số cổng nào đó(anonymous) và thường được  
sử dụng phía chương trình client. Nếu tạo socket không thành công, nó ném trả về  
ngoại lệ SocketException.  
**Ví dụ 2 :**

|  |
| --- |
| try { DatagramSocket client = new DatagramSocket( ); // send packets... } catch (SocketException ex) {  System.err.println(ex) ;  } |

•public DatagramSocket(int port) throws SocketException:  
Hàm tạo này cho phép tạo socket với số cổng xác định và chờ nhận gói tín truyền tới.  
Hàm tạo này được sử dụng phía server trong mô hình client/server.  
Ví dụ 3 : Chương trình sau sẽ cho phép hiển thị các cổng cục bộ đã được sử dụng:

|  |
| --- |
| import java.net.\*; public class UDPPortScanner { public static void main(String[] args) { for (int port = 1024; port <= 65535; port++) { try { // the next line will fail and drop into the catch block if // there is already a server running on port i DatagramSocket server = new DatagramSocket(port); server.close( ); } catch (SocketException ex) { System.out.println( "There is a server on port " + port + "."); } // end try } // end for }} |

Các phương thức  
•public void send(DatagramPacket dp) throws IOException:  
Phương thức này cho phép gửi gói tin UDP qua mạng.  
Chương trình sau nhận một chuỗi từ bàn phím, tạo gói tin gửi và gửi tới server.

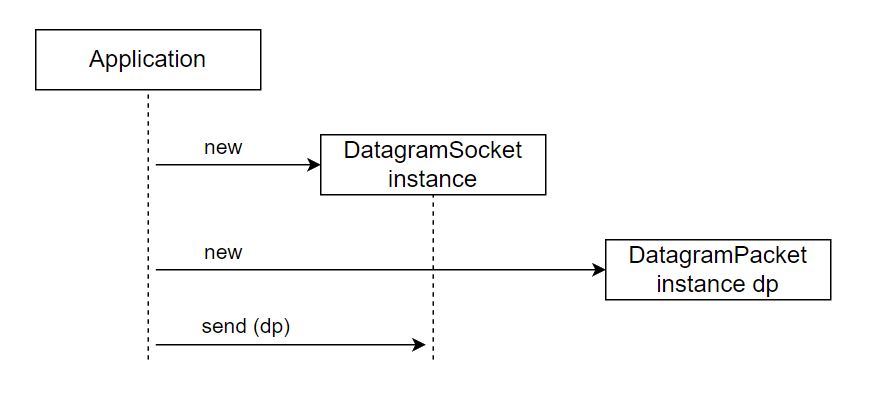
|  |
| --- |
| import java.net.\*; import java.io.\*; public class UDPDiscardClient { public final static int DEFAULT PORT = 9; public static void main(String[] args) { String hostname; int port = DEFAULT\_PORT;  if (args.length > 0) { hostname = args[0]; try { port = Integer.parseInt(args[1]); } catch (Exception ex) { } } else { hostname = "localhost"; } try { InetAddress server = InetAddress.getByName(hostname); BufferedReader userInput= new BufferedReader(InputStreamReader(System.in)); DatagramSocket theSocket = new DatagramSocket( ); while (true) { String theLine = userInput.readLine( ); if (theLine.equals(".")) break; byte[] data = theLine.getBytes( ); DatagramPacket theOutput= new DatagramPacket(data, data.length, server, port); theSocket.send (theOutput); } // end while } // end try catch (UnknownHostException uhex) {System.err.println(uhex); } catch (SocketException sex) { System. err.println (sex) ; } catch (IOException ioex) { System.err.println (ioex) ; } } // end main} |

•public void receive(DatagramPacket dp) throws IOException:  
Phương thức nhận gói tin UDP qua mạng.  
•public void close(): Phương thức đóng socket.  
Ví dụ 4 : Chương trình sau sẽ tạo đối tượng DatagramSocket với số cổng xác định,  
nghe nhận gói dữ liệu gửi đến, hiển thị nội dung gói tin và địa chỉ, số cổng của máy trạm gửi gói tin.

|  |
| --- |
| import java.net.\*; import java.io.\*; public class UDPDiscardServer { public final static int DEFAULT PORT = 9; public final static int MAX PACKET SIZE = 65507; public static void main(String[] args) { int port = DEFAULT\_PORT; byte[] buffer = new byte[MAX PACKET SIZE]; try { port = Integer.parseInt(args[0]); } catch (Exception ex) { // use default port } try { DatagramSocket server = new DatagramSocket(port); DatagramPacket packet = new DatagramPacket(buffer, buffer.length); while (true) { try { server.receive(packet) ; String s = new String(packet.getData( ), 0 , packet.getLength( )); System.out.println(packet.getAddress( )+ " at port "+ packet.getPort( )+ " says " + s); // reset the length for the next packet packet.setLength(buffer.length); } catch (IOException ex) {  System.err.println(ex) ; } } // end while }// end try catch (SocketException ex) { System.err.println(ex); } // end catch } // end main } |

**II. Kỹ thuật lập trình truyền thống với giao thức UDP**

|  |  |
| --- | --- |
| Một số phương thức của lớp DatagramSocket | |
| void | bind (SocketAddress addr) Gắn kết DatagramSocket với địa chỉ và số cổng cụ thể |
| void | connect( InetAddress a d d r e s s , i n t p o r t ) Kết nối socket với địa chỉ máy trạm từ xa |
| void | connect ( SocketAddress addr ) Kết nối socket với địa chỉ socket từ xa. |
| void | disconnect ( ) Huỷ bỏ kết nối |
| boolean | isBound ( ) Trả về trạng thái kết nối của socket. |
| boolean | isCỉosed ( ) Kiểm tra socket đã đóng hay chưa |
| boolean | isConnected ( ) Kiểm tra trạng thái kết nối |

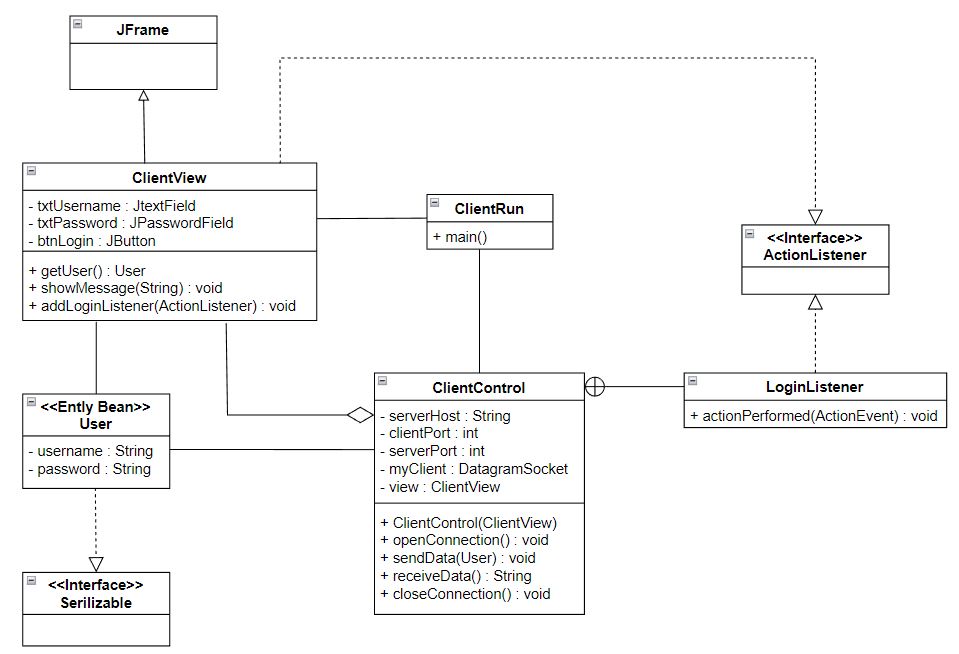


Hình 2.1 Quá trình khởi tạo truyền thông UDPSocketTrong mô hình client/server, để chương trình client và server có thể truyền thông đượcvới nhau, mỗi phía phải thực hiện một số thao tác cơ bản sau đây (Hình 6.2)- Phía server  
•Tạo đối tượng DatagramSocket với số cổng cho trước  
•Khai báo bộ đệm nhập /xuất inBuffer/outBuffer dạng mảng kiểu byte  
•Khai báo gói tin nhận gửi inData/outData là đối tượng DatagramPacket.  
•Thực hiện nhận/gửi gói tin với phương thức receive()/send()  
•Đóng socket, giải phóng các tài nguyên khác, kết thúc chương trình nếu  
cần, không quay về bước 3.  
- Phía client  
•Tạo đối tượng DatagramSocket với số cổng cho trước  
•Khai báo bộ đệm xuất/nhập outBuffer/inBuffer dạng mảng kiểu byte  
•Khai báo gói tin gửi/nhận outData/inData là đối tượng DatagramPacket.  
•Thực hiện gửi /nhận gói tin với phương thức send()/receive()  
•Đóng socket, giải phóng các tài nguyên khác, kết thúc chương trình nếu cần,  
không quay về bước 3.  
Một số chương trình ví dụ lập trình giao thức UDPVí dụ 5 : Chương trình minh hoạ sử dụng giao thức UDP để truyền dữ liệu

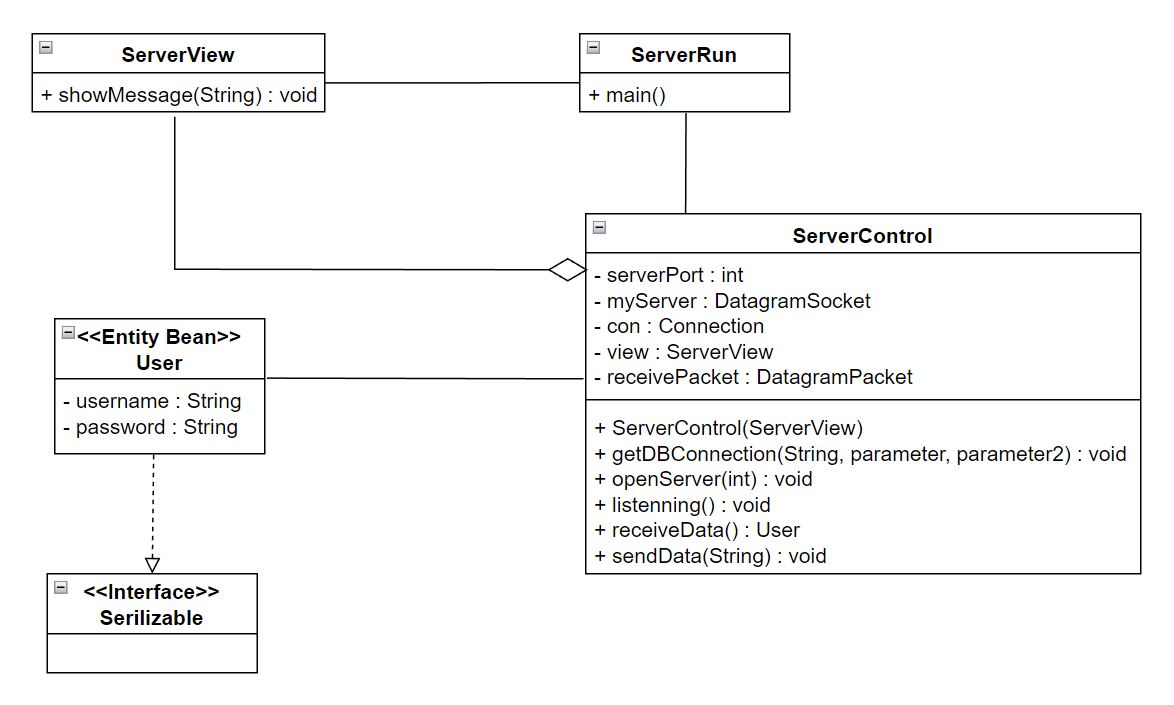
|  |
| --- |
| //UDPEchoClient.java import java.net.\*' import java.io.\*; public class UDPEchoClient { public final static int DEFAULT PORT = 7; public static void main(String[] args) { String hostname = "localhost"; int port = DEFAULT\_PORT; if (args.length > 0) { hostname = args[0]; } try { InetAddress ia = InetAddress.getByName(hostname); Thread sender = new SenderThread(ia, DEFAULT PORT); sender.start( ); Thread receiver = new ReceiverThread(sender.getSocket( )); receiver.start( ); }  catch (UnknownHostException ex) {catch (SocketException ex) { System.err.println (ex) ; } } // end main } //UDPEchoServer.java import java.net.\*; import java.io.\*; public class extends UDPServer { public final static int DEFAULT PORT = 7; public UDPEchoServer( ) throws SocketException { super (DEFAULT\_PORT) ; } public void respond(Datagram.Packet packet) { try { DatagramiPacket outgoing = new DatagramPacket(packet.getData( ), packet.getLength( ), packet.getAddress( ), packet.getPort( )); socket.send(outgoing); } catch (IOException ex) {System.err.println (ex) ; } } public static void main(String[] args) { try { UDPServer server = new UDPEchoServer( ); server.start( ); } catch (SocketException ex) {System.err.println(ex); }  }  } |

Ví dụ 6 : **Xây dựng chương trình** Login từ xa dùng giao thức UDPBài toán login từ xa dùng giao thức UDP đặt ra như sau:  
- Cơ sở dữ liệu được lưu trữ và quản lí trên server UDP, trong đó có bảng users chứa  
ít nhất hai cột: cột username và cột password.  
- Chương trình phía client UDP phải hiện giao diện đồ họa, trong đó có một ô text để  
nhập username, một ô text để nhập password, và một nút nhấn Login.  
- Khi nút Login được click, chương trình client sẽ gửi thông tin đăng nhập  
(username/password) trên form giao diện, và gửi sang server theo giao thức UDP  
- Tại phía server, mỗi khi nhận được thông tin đăng nhập gửi từ client, sẽ tiến hành  
kiểm tra trong cơ sở dữ liệu xem có tài khoản nào trùng với thông tin đăng nhập  
nhận được hay không.Sau khi có kết quả kiểm tra (đăng nhập đúng, hoặc sai),  
server UDP sẽ gửi kết quả này về cho client tương ứng, theo đúng giao thức UDP.  
- Ở phía client, sau khi nhận được kết quả đăng nhập (đăng nhập đúng, hoặc sai) từ  
server, sẽ hiển thị thông báo tương ứng với kết quả nhận được: nếu đăng nhập đúng  
thì thông báo login thành công. Nếu đăng nhập sai thì thông báo là username/password không đúng. Yêu cầu kiến trúc hệ thống được thiết kế theo mô hình MVC  
Kiến trúc hệ thống theo mô hình MVCSơ đồ lớp của phía client được thiết kế theo mô hình MVC bao gồm 3 lớp chính tương ứng với sơ đồ M-V-C như sau:

Hình 2.2 Sơ đồ lớp phía client UDP  
- Lớp User: là lớp tương ứng với thành phần model (M), bao gồm hai thuộc tính  
username và password, các hàm khởi tạo và các cặp getter/setter tương ứng với các  
thuộc tính.  
- Lớp ClientView: là lớp tương ứng với thành phần view (V), kế thừa từ lớp  
JFrame của Java, chứa các thuộc tính là các thành phần đồ họa bao gồm ô text nhập  
username, ô text nhập password, nút nhất Login.  
- Lớp ClientControl: là lớp tương ứng với thành phần control (C), chứa một lớp  
nội tại là LoginListener. Khi nút Login trên tầng view bị click thì nó sẽ chuyển tiếp sự  
kiện xuống lớp nội tại này để xử lí.  
- Tất cả các xử lí đều gọi từ phương thức actionPerformed của lớp nội tại này,  
bao gồm: lấy thông tin trên form giao diện và gửi sang server theo giao thức UDP,  
nhận kết quả đăng nhập từ server về và yêu cầu form giao diện hiển thị. Điều này đảm  
bảo nguyên tắc control điều khiển các phần còn lại trong hệ thống, đúng theo nguyên  
tắc của mô hình MVC.



Hình 2.3 Sơ đồ lớp phía server UDPSơ đồ lớp của phía server được thiết kế theo mô hình MVC trong Hình 6.7, bao  
gồm 3 lớp chính tương ứng với sơ đồ M-V-C như sau:  
- Lớp User: là lớp thực thể, dùng chung thống nhất với lớp phía bên client.  
- Lớp ServerView: là lớp tương ứng với thành phần view (V), là lớp dùng hiển  
thị các thông báo và trạng thái hoạt động bên server UDP.  
- Lớp ServerControl: là lớp tương ứng với thành phần control (C), nó đảm nhiệm  
vai trò xử lí của server UDP, bao gồm: nhận thông tin đăng nhập từ phía các client,  
kiểm tra trong cơ sở dữ liệu xem các thng tin này đúng hay sai, sau đó gửi kết quả  
đăng nhập về cho client tương ứng.



❖Lớp phía client User.java

|  |
| --- |
| package udp.client; 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; } } |

Client View. java

|  |
| --- |
| package udp.client; import java.awt.FlowLayout; import j ava.awt.event.ActionEvent; import java.awt.event.ActionListener; import java.awt.event.WindowAdapter; import java.awt.event.WindowEvent; import javax.swing.\*; public class ClientView extends JFrame implements ActionListener{ private JTextField txtUsername; private JPasswordField txtPassword; private JButton btnLogin; 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()); c ontent.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); } }); } 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); }  } |

ClientControl.java

|  |
| --- |
| package udp.client; 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; 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(); } } |

|  |
| --- |
| private void openConnection(){ try { myClient = new DatagramSocket(clientPort); } catch (Exception ex) { view.showMessage(ex.getStackT race().toString()); } } private void closeConnection() { try { myClient.close(); } catch (Exception ex) { view.showMessage(ex.getStackT race().toString()); } } 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.showMessage(ex.getStackT race().toString()); } } 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.showMessage(ex.getStackT race().toString()); } return result; } } |

ClientRun.java

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

❖Các lớp phía server

|  |
| --- |
| package udp.server; public class ServerView { public ServerView(){ } public void showMessage(String msg){ System.out.println(msg); } } |

ServerControl.java

|  |
| --- |
| package udp.server; import java.io.ByteArrayInputStream; import j ava.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; public ServerControl(ServerView view){ this.view = view; getDBConnection("usermanagement", "root", "12345678"); openServer(serverPort); view.showMessage("UDP server is running..."); while(true){ listenning(); } } private void getDBConnection(String dbName, String username, String password) { String dbUrl = "jdbc:mysql://localhost:3306/" + 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 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); } 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.showMessage(ex.getStackT race().toString()); } } 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.readObjjctQ; } catch (Exception ex) { view.showMessage(ex.getStackT race().toString()); } return user; } 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.showMessage(e.getStackTrace().toString()); } return false; } } |

ServerRun.java

|  |
| --- |
| package udp.server; public class ServerRun { public static void main(String[] args) {  ServerView view = new ServerView(); ServerControl control = new ServerControl(view); } |