TÀI LIỆU THỰC TẬP LẬP TRÌNH MẠNG: LAB 10

DANH MỤC THUẬT NGỮ TIẾNG ANH

Từ	Nghĩa của từ
abstract	Trừu tượng
break	Dừng vòng lặp
catch	Từ khóa đầu của một khối bắt ngoại lệ
continue	Bỏ qua phần cuối vòng lặp, tiếp tục sang bước tiếp theo
default	Giá trị mặc định của phương thức switch()
extends	Kế thừa
final	Một hằng số, phương thức hay một lớp không được ghi đè
finally	Một phần của khối xử lý ngoại lệ try luôn được thực hiện
implements	Thực hiện giao diện
import	Khai báo một gói thư viện
instanceof	Kiểm tra một đối tượng là một thể hiện của lớp
interface	Giao diện
new	Tạo một đối tượng mới của lớp
null	Tham chiếu rỗng
package	Gói
private	Tiền tố chỉ được truy cập bởi phương thức của lớp
protected	Tiền tố được truy cập bởi phương thức của lớp, lớp con của và các lớp khác trong cùng một gói
public	Tiền tố có thể được truy cập bởi phương thức của tất cả các lớp
return	Trả về của một phương thức
super	Gọi phương thức của lớp cha
synchronized	Đồng bộ
this	Tham chiếu đến đối tượng hiện tại

DANH MỤC CHỮ VIẾT TẮT

Chữ viết tắt	Ý nghĩa
UDP	User Datagram Protocol
TCP	Transmission Control Protocol
IP	Internet Protocol
URL	Uniform Resource Locator
CSDL	Cơ Sở Dữ Liệu
JDBC	Java Database Connectivity
CNTT	Công Nghệ Thông Tin
HĐH	Hệ Điều Hành
MVC	Model-View-Control
DNS	Domain Name System
API	Application Programming Interface
FTP	File Transfer Protocol
JDK	Java Development Kit
GB	GigaByte
UCLN	Ước Chung Lớn Nhất
BCNN	Bội Chung Nhỏ Nhất
RAM	Random Access Memory
RMI	Remote Method Invocation
JVM	Java Virtual Machine
NIC	Network Interface Card
ĐH KTKT CN	Đại học Kinh tế Kỹ thuật Công nghiệp

LỜI NÓI ĐẦU

Ngày nay do nhu cầu thực tế và do sự phát triển mạnh mẽ của nhiều công nghệ tích hợp, dẫn đến các chương trình ứng dụng hầu hết đều có khả năng thực hiện trên môi trường mạng. Ngôn ngữ JAVA là ngôn ngữ phù hợp để viết các ứng dụng mạng. So với lập trình thông thường, lập trình mạng đòi hỏi người lập trình hiểu biết và có kỹ năng tốt để viết các chương trình giao tiếp và trao đổi dữ liệu giữa các máy tính với nhau.
Để hỗ trợ sinh viên chuyên ngành CNTT trong nhà trường tiếp cận với kỹ thuật lập trình mới này, tiếp theo cuốn tài liệu học tập lý thuyết "**Công nghệ JAVA**", chúng tôi xây dựng cuốn "**Bài tập lập trình mạng**", nhằm cung cấp cho sinh viên những kiến thức và kỹ thuật cơ bản nhất để phát triển các chương trình ứng dụng mạng, thông qua các dạng bài tập từ cơ bản đến nâng cao qua các chủ đề: lập trình cơ bản, lập trình hướng đối tượng, lập trình CSDL JDBC, lập trình mạng dùng socket, lập trình phân tán với RMI. Sinh viên sẽ thực hiện các bài thực hành này trên phòng máy nhà trường.

Nội dung cuốn tài liệu bao gồm 12 bài lab chia thành các chủ đề khác nhau. Trong mỗi chủ đề chúng tôi đưa ra tóm tắt lý thuyết, bài tập mẫu, sau đó là bài tập tương tự, và bài tập tổng hợp. Kết quả qua những bài lab, sinh viên được rèn và thành thạo các kỹ năng lập trình hướng đối tượng, lập trình CSDL, lập trình với giao thức truyền thông có sẵn và khả năng tích hợp trong các ứng dụng khác nhau, nhất là các giao thức truyền thông thời gian thực, từ đó sinh viên có thể viết được các phần mềm quản lý theo mô hình MVC, xây dựng được các ứng dụng mạng, các ứng dụng tích hợp và triệu gọi lẫn nhau trên mạng Intranet (mạng cục bộ), mạng Internet (mạng toàn cầu), các hệ thống xử lý truy xuất dữ liệu phân tán hoàn chỉnh. Nội dung biên soạn phù hợp với chuẩn đầu ra của ngành CNTT và ngành mạng máy tính và truyền thông dữ liệu về kỹ năng và kiến thức. Sau khi học xong học phần này sinh viên có thể viết phần mềm quản lý, truyền thông.

Chúng tôi xin chân thành cảm ơn Thầy Nguyễn Hoàng Chiến, phó chủ nhiệm khoa, phụ trách khoa CNTT trường ĐH KTKT CN cùng với các đồng nghiệp đã đóng góp ý kiến cho cuốn tài liệu này. Vì tài liệu được biên soạn lần đầu, chúng tôi đã cố gắng hoàn chỉnh, song không tránh khỏi thiếu sót. Rất mong nhận được sự góp ý của bạn đọc để tài liệu học tập được hoàn thiện hơn.

Xin trân trọng cảm ơn!

Nhóm tác giả

LAB 10. LẬP TRÌNH CLIENT-SERVER SỬ DỤNG UDP [4, 6, 9,10]

A. MỤC TIÊU

Trang bị cho sinh viên kỹ thuật lập trình Client/Server sử dụng lớp DatagramSocket, DatagramPackage, để kết nối và trao đổi thông tin bằng giao thức UDP.

B. NỘI DUNG

Lập trình Client/Server sử dụng lớp DatagramSocket, DatagramPackage.

C. KẾT QUẢ SAU KHI HOÀN THÀNH

Viết được các ứng dụng như: chat đa người dùng.

D. YÊU CẦU PHẦN CỨNG - PHẦN MỀM

- Máy tính cài HĐH windows, RAM tối thiểu 1GB, có kết nối Internet.
- Phần mềm NETBEAN 8.0, JDK 1.8. MYSQL.

E. HƯỚNG DẪN

1. Giao thức hướng không kết nối – UDP: thực hiện truyền thông không cần kết nối (ảo)

Đặc điểm: Truyền thông theo kiểu điểm-đa điểm

- Quá trình truyền thông chỉ có một giai đoạn duy nhất là truyền dữ liệu, không có giai đoạn thiết lập cũng như huỷ bỏ kết nối.
- Dữ liệu truyền được tổ chức thành các tin gói tin độc lập, trong mỗi gói dữ liệu có chứa địa chỉ nơi nhận.

2. Một số lớp dùng trong lập trình UDP

a. Lớp DatagramPacket cho phép tạo gói tin truyền thông với giao thức UDP.

Gói tin 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 phương thức tạo

public DatagramPacket(**byte**[] inBuffer, **int** length): Tạo gói tin nhận từ mạng.

Tham số:

- inBuffer: Bộ đệm nhập, chứa dữ liệu của gói tin nhận
- length: Kích cỡ của dữ liệu của gói tin nhận

public DatagramPacket(**byte**[] outBuffer, **int** length, **InetAddress** destination, **int** port): Tạo gói tin gửi.

Tham số:

- outBuffer: Bộ đệm xuất chưa dữ liệu của gói tin gửi
- length: Kích cỡ dữ liệu của gói tin gửi (byte)
- destination: Địa chỉ nơi nhận gói tin.
- port: Số hiệu cổng đích, nơi nhận gói tin.

Các phương thức

- **public** InetAddress getAddress(): Trả về đối tượng InetAddress của máy trạm từ xa chứa trong gói tin nhận.

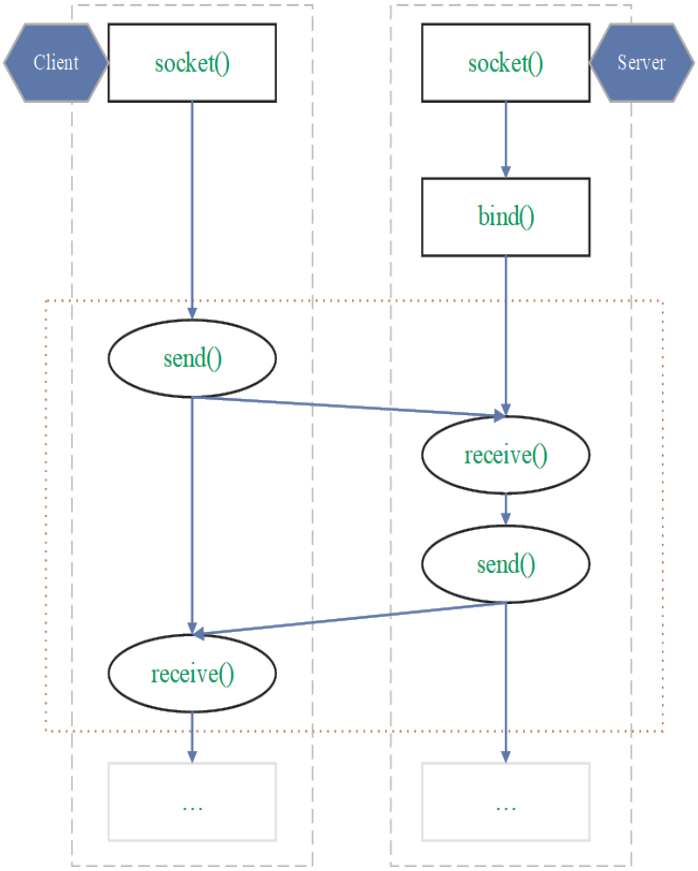
- **public int** getPort(): Trả về số hiệu cổng của máy trạm từ xa chứa trong gói tin.
- **public** byte[] getData() : Trả về dữ liệu chứa trong gói tin dưới dạng mảng byte.
- **public int** getLength(): Trả về kích cỡ của dữ liệu chứa trong gói tin

b. Lớp DatagramSocket

Tạo đối tượng socket truyền thông với giao thức UDP. Socket này cho phép gửi/nhận gói tin DatagramPacket.

Các phương thức tạo

- **public** DatagramSocket() Sử dụng phía Client tạo ra socket với số cổng nào đó.
- **public** DatagramSocket(**int** port) Sử dụng phía Server trong mô hình Client/Server, tạo socket với số cổng xác định và chờ nhận gói tín truyền tới.



Hình 10. Mô hình Cient Server ở chế độ không kết nối

3. Các bước lập trình Client-Server sử dụng UDP

a) Phía Server

1. Tạo đối tượng DatagramSocket với số cổng xác định
2. Khai báo bộ đệm nhập /xuất inBuffer/outBuffer dạng mảng kiểu byte
3. Khai báo gói tin nhận gửi inData/outData là đối tượng DatagramPacket.
4. Thực hiện nhận/gửi gói tin với phương thức receive()/send()
5. Đóng socket, giải phóng tài nguyên, kết thúc chương trình, nếu không quay về bước 3.

b) Phía Client

1. Tạo đối tượng DatagramSocket với số cổng nào đó

2. Khai báo bộ đệm xuất/nhập outBuffer/inBuffer dạng mảng kiểu byte
3. Khai báo gói tin gửi/nhận outData/inData là đối tượng DatagramPacket.
4. Thực hiện gửi /nhận gói tin với phương thức send()/receive()
5. Đóng socket, giải phóng tài nguyên, kết thúc chương trình, nếu không quay về bước 3.

Bài 1.

Viết chương trình Client-Server minh họa sử dụng DatagramPacket. Phía Client gửi tin nhắn trong DatagramPacket đến máy chủ. Máy chủ lắng nghe trên cổng UDP. Nếu máy chủ nhận gói dữ liệu datagram, nó sẽ in thông điệp trong gói datagram để hiển thị.

Hướng dẫn

Bước 1: Tạo class Client

```
import java.net.*;
import java.io.*;
public class Client extends java.applet.Applet {
	String myHost;
	public void init() {
		myHost = getCodeBase().getHost();
	}
	public void sendMessage( String message ) {
		//phương thức gửi tin nhắn
		try {
			byte[] data = new byte[ message.length() ];
			message.getBytes(0, data.length, data, 0);
			InetAddress addr = InetAddress.getByName(myHost);
DatagramPacket pack = new DatagramPacket(data, data.length, addr,1234);
			DatagramSocket ds = new DatagramSocket();
			ds.send(pack);
			ds.close();
		}
		catch (IOException e) {
			System.out.println( e );
		}
	}
	public void start() {
		sendMessage( "Bắt đầu!" );
	}
	public void stop() {
		sendMessage( "Kết thúc!" );
	}
}
```

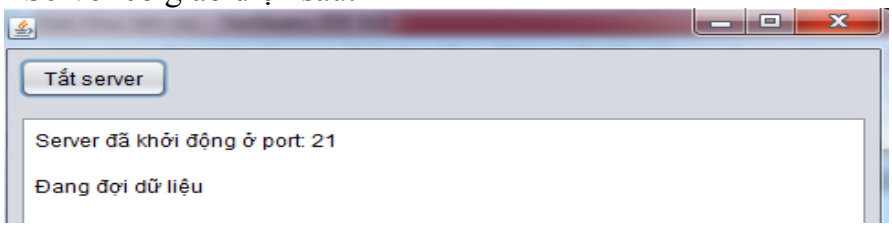
Bước 2: Tạo class Server

```
import java.net.*;
import java.io.*;
public class Server{
     public static void main( String argv[] ) throws IOException {
          DatagramSocket s = new DatagramSocket( 1234 );
          System.out.println(" Lắng nghe..");
          while (true) {
     DatagramPacket packet = new DatagramPacket( new byte[1024], 1024);
               s.receive( packet );
String message = new String(packet.getData(), 0, 0, packet.getLength());
   System.out.println( "Tín hiệu từ "+packet.getAddress().getHostName()+"
- "+message );
          }
     }
}
```

Bài 2. Viết chương trình Client-Server xử lý chuỗi, sử dụng giao thức UDP: Phía Client gửi một chuỗi lên Server, Server xử lý chuyển chữ hoa và đếm số từ trong xâu gửi lên.

Hướng dẫn:

Tạo màn hình Server có giao diện sau:



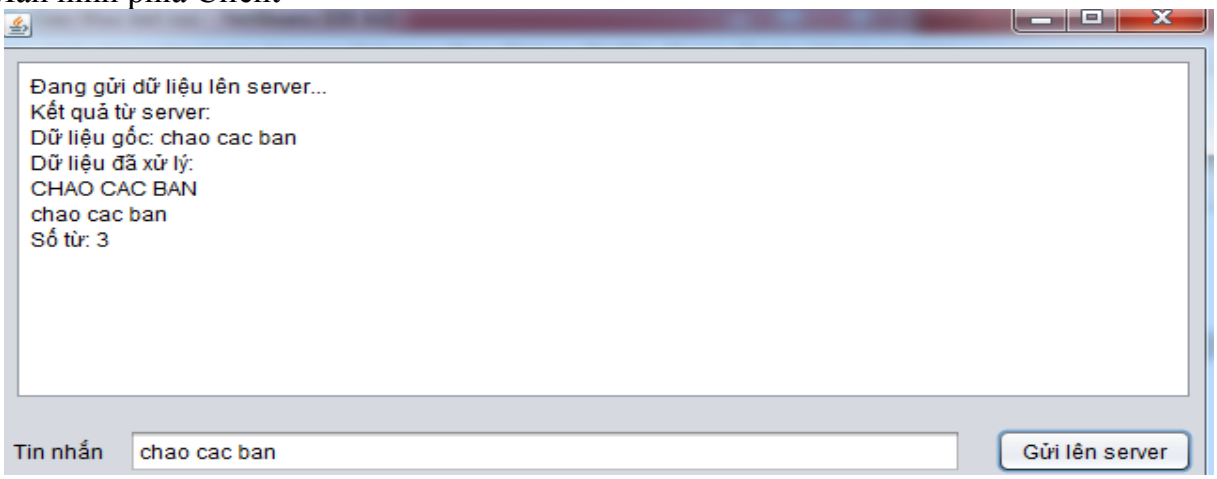
Bước 1: Tạo Class UDPServer kế thừa từ Jframe, có btnBatTat

UDPServer **extends** javax.swing.Jframe

Viết phương thức: **private void** btnBatTatActionPerformed(java.awt.event.ActionEvent evt)

//Xử lý sự kiện bật/tắt Server.

Màn hình phía Client



Bước 2: Tạo Class UDPClient có giao diện như trên viết phương thức xử lý sự kiện cho btnSendToServer

```
private void btnSendToServerActionPerformed(java.awt.event.ActionEvent
evt) {
        txaStatus.append("Đang gửi dữ liệu lên Server...\n");
        int port = 21;
        String hostName = "localhost";
        byte[] outBuf = new byte[2048];
        byte[] inBuf = new byte[2048];
        try
        {
            //khai báo địa chỉ để kết nối
            InetAddress address = InetAddress.getByName(hostName);
            DatagramSocket socket = new DatagramSocket();
            //chuyển đổi tin nhắn gửi đi sang byte
            outBuf = txtTinNhan.getText().getBytes();
            //Gửi dữ liệu đi
            DatagramPacket outPacket = new DatagramPacket(outBuf, 0,
outBuf.length, address, port);
            socket.send(outPacket);
            //Nhận dữ liệu về
  DatagramPacket receivePacket = new DatagramPacket(inBuf, inBuf.length);
            socket.receive(receivePacket);
   String receiveMsg = new String(receivePacket.getData(),0,
receivePacket.getLength());
            txaStatus.append("Kết quả từ Server:\n" + receiveMsg);
        } catch (IOException e)
            {
                e.printStackTrace();
            }
    }
```

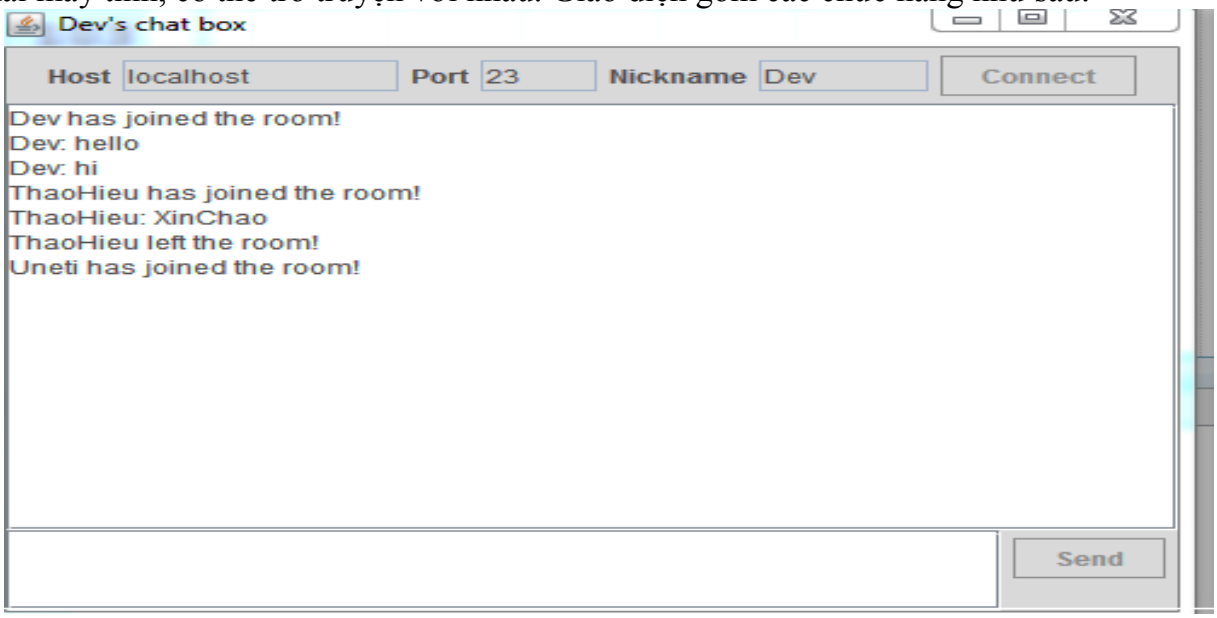
Bước 3: Tạo class UDPServer_Thread

```
public class UDPServer_Thread extends Thread
  {
    DatagramSocket mServerSocket;
    JTextArea mTxaStatus;    //JTextArea để lưu status của Server
    public UDPServer_Thread(JTextArea txaStatus)
    {
        mTxaStatus = txaStatus;
    }
```

```
        @Override
    public void run()
    {
        int port = 21;
             try
                {
            mServerSocket = new DatagramSocket(port);
  mTxaStatus.append("Server khởi động ở: "+port+"Đang đợi dữ liệu\n");
            byte[] buf = new byte[2048];
     DatagramPacket receivePacket = new DatagramPacket(buf, buf.length);
            while (true)
            {
              try
                {
                    mServerSocket.receive(receivePacket);
          String ClientMsg = new String(receivePacket.getData(), 0,
     receivePacket.getLength());
       StringTokenizer st = new StringTokenizer(ClientMsg, " ");//đếm số
    từ dựa vào dấu space
       int numOfWords = st.countTokens();
       String returnMsg = "Dữ liệu gốc: " + ClientMsg + "\n"
             + "Dữ liệu đã xử lý:\n"+ ClientMsg.toUpperCase() + "\n"
    + ClientMsg.toLowerCase() + "\n" + "Số từ: "  + numOfWords + "\n\n";
                    mTxaStatus.append(returnMsg);
DatagramPacket outPacket = new DatagramPacket(returnMsg.getBytes(),
returnMsg.getBytes().length,receivePacket.getAddress(),
receivePacket.getPort());
                    mServerSocket.send(outPacket);
                } catch (IOException e) {
                        e.printStackTrace();
                }
            }
        } catch (SocketException e) {
                e.printStackTrace();
        }
    }
 public void StopServer()
    {
        super.stop();
        mServerSocket.close();
```

```
    }
}
```

Bài 3. Viết chương trình tán gẫu (chat) theo chế độ không nối kết. Cho phép hai người ở hai máy tính, có thể trò truyện với nhau. Giao diện gồm các chức năng như sau:



Bước 1: Tạo class ChatUDPServer

```
import java.io.IOException;
import java.net.*
import java.util.*;
public class ChatUDPServer {
	static int port = 23;//Mở cổng 23
	static byte[] inBuf = new byte[2048];
	static DatagramSocket socket;
	static ArrayList<User> inUsers = new ArrayList<User>();
	static class User {
		private InetAddress address;
		private int port;
		public User(InetAddress address, int port) {
			super();
			this.address = address;
			this.port = port;
		}
		public InetAddress getAddress() {
			return address;
		}
		public void setAddress(InetAddress address) {
			this.address = address;
		}
```

```
            public int getPort() {
                  return port;
            }
            public void setPort(int port) {
                  this.port = port;
            }
        }
      public static void main(String[] args) {
            try {
                  socket = new DatagramSocket(port);
                  System.out.println("Server started at port " + port);
      DatagramPacket inPacket = new DatagramPacket(inBuf, inBuf.length);
                  while (true) {
                        try {
                              socket.receive(inPacket);
                              boolean firstJoin = true;
      User user = new User(inPacket.getAddress(), inPacket.getPort());
                  System.out.println("User port: " + user.getPort());
                  for (User u : inUsers) {
      if (u.getPort() == user.getPort() &&
u.getAddress().equals(user.getAddress())) {
                                          firstJoin = false;
                                          }
                                        }
      System.out.println("inPackets length: " + inUsers.size());
String inMsg = new String(inPacket.getData(), 0, inPacket.getLength());
                              // Gửi tín hiệu tới phòng chat
                  StringTokenizer st = new StringTokenizer(inMsg, "\t");
                              String senderName = st.nextToken();
                              String msg = st.nextToken();
                              if (msg.equals("~leave")) {
                                    // Người dùng rời phòng chat
                                    for (User u : inUsers) {
if (u.getAddress().equals(user.getAddress()) && u.getPort() ==
user.getPort())             {
                                     user = u;
                                     }
                              }
                    inUsers.remove(user);
                    sendMsg(senderName + " left the room!\n");
```

```
                        } else {
                    if (firstJoin) {
                    inUsers.add(user);
                    sendMsg(senderName + " has joined the room!\n");
                    firstJoin = false;
                              }
                              else {
                    String outMsg = senderName + ": " + msg + "\n";
                                    System.out.println(outMsg);
                                    sendMsg(outMsg);
                              }
                        }
                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                  }
              } catch (SocketException e) {
                e.printStackTrace();
          }
      }
private static void sendMsg(String outMsg) {
DatagramPacket outPacket = new DatagramPacket(outMsg.getBytes(),
outMsg.length(), null, 0);
          for (User user : inUsers) {
                outPacket.setAddress(user.getAddress());
                outPacket.setPort(user.getPort());
                if (socket != null) {
                      try {
                            socket.send(outPacket);
System.out.println("Sent to chatroom: " +
outPacket.getAddress().toString() + " " + outPacket.getPort() + "\n" +
new String(outPacket.getData(), 0, outPacket.getLength()));
                    } catch (IOException e) {
                          e.printStackTrace();
                    }
                }
          }
      }
}
```

Bước 2: Tạo class UDPClient

```
import java.awt.*;
import java.io.IOException;
import java.net.*;
import javax.swing.*
public class ChatUDPClient extends JFrame implements ActionListener {
	static String nickName = "Dev";
	static String hostName = "localhost";
	static int port = 23;
	static DatagramSocket socket;
	InetAddress address = null;
	static JTextArea chatInput = null;
	static JTextArea chatView = null;
	JTextField hostTF;
	JTextField portTF;
	JTextField nickNameTF;
	JButton sendBtn;
	JButton connBtn;
	static Thread getMsgThread;
	public ChatUDPClient() {
		initFrame();
	       }
	public static void main(String[] args) {
		new ChatUDPClient();
		getMsgThread = new Thread(new MsgReceiver());
	}
	static class MsgReceiver implements Runnable {
		byte[] buf = new byte[2048];
		@Override
		public void run() {
			while (true) {
	DatagramPacket inPacket = new DatagramPacket(buf, buf.length);
				try {
					socket.receive(inPacket);
					if (inPacket.getLength() != 0) {
					}
String inMsg = new String(inPacket.getData(), 0,inPacket.getLength());
					chatView.append(inMsg);
				} catch (IOException e) {
					e.printStackTrace();
				}
```

```
                }
            }
        }
 // Phương thức tạo giao diện
        private void initFrame() {
            setSize(500, 400);
            setMinimumSize(new Dimension(500, 400));
            setTitle(nickName + "'s chat box");
            Container container = getContentPane();
            JPanel panel = new JPanel(new BorderLayout());
            JPanel topPanel = new JPanel();
            JLabel lb1 = new JLabel("Host");
            hostTF = new JTextField(10);
            JLabel lb2 = new JLabel("Port");
            portTF = new JTextField(4);
            JLabel lb3 = new JLabel("Nickname");
            nickNameTF = new JTextField(6);
            connBtn = new JButton("Connect");
            connBtn.addActionListener(this);
            hostTF.setText(hostName);
            portTF.setText(port + "");
            nickNameTF.setText(nickName);
            topPanel.add(lb1);
            topPanel.add(hostTF);
            topPanel.add(lb2);
            topPanel.add(portTF);
            topPanel.add(lb3);
            topPanel.add(nickNameTF);
            topPanel.add(connBtn);
            panel.add(topPanel, "North");
            chatView = new JTextArea();
            chatView.setEditable(false);
            chatView.setLinewrap(true);
            JScrollPane conversationScrollView = new
JScrollPane(chatView);
            JScrollPane chatInputScrollView = new JScrollPane();
            panel.add(conversationScrollView, "Center");
            chatInput = new JTextArea(3, 10);
            chatInput.setLinewrap(true);
            chatInput.setEditable(false);
```

```
chatInput.getDocument().addDocumentListener(new DocumentListener()
{
	@Override
			public void removeUpdate(DocumentEvent e) {
					checkInput();
			}
		@Override
			public void insertUpdate(DocumentEvent e) {
				checkInput();
			}
		});
		chatInputScrollView.setViewportView(chatInput);
		JPanel chatToolPanel = new JPanel(new BorderLayout());
		chatToolPanel.add(chatInputScrollView, "Center");
		sendBtn = new JButton("Send");
		sendBtn.setEnabled(false);
		sendBtn.addActionListener(this);
		JPanel btnPanel = new JPanel();
		btnPanel.add(sendBtn);
		chatToolPanel.add(btnPanel, "East");
		panel.add(chatToolPanel, "South");
		container.add(panel);
		setDefaultCloseOperation(EXIT_ON_CLOSE);
	Runtime.getRuntime().addShutdownHook(new Thread(new Runnable() {
			public void run() {
			  leave();
			}
		}));
		setVisible(true);
	    }
	private void leave() {
		try {
			sendMsg(nickName + "\t" + "~leave");
		} catch (IOException e) {
			e.printStackTrace();
		}
	  }
	private void checkInput() {
		if (chatInput.getText().length() > 0) {
			sendBtn.setEnabled(true);
```

```
            }
            else {
                  sendBtn.setEnabled(false);
            }
      }
      @Override
      public void actionPerformed(ActionEvent e) {
            String btnName = ((JButton) e.getSource()).getText();
            switch (btnName) {
            case "Connect":
                  nickName = nickNameTF.getText();
                  hostName = hostTF.getText();
                  port = Integer.parseInt(portTF.getText());
                  try {
                        socket = new DatagramSocket();
                        address = InetAddress.getByName(hostName);
                        String msg = nickName + "\t" + "join";
                        sendMsg(msg);
                        getMsgThread.start();
                        chatInput.setEditable(true);
                        this.setTitle(nickName + "'s chat box");
                        hostTF.setEditable(false);
                        portTF.setEditable(false);
                        nickNameTF.setEditable(false);
                        connBtn.setEnabled(false);
                  } catch (IOException e2) {
                        e2.printStackTrace();
                  }
                  break;
            case "Send":
                  String msg = nickName + "\t" + chatInput.getText();
                  try {
                        sendMsg(msg);
                    } catch (IOException e1) {
                        e1.printStackTrace();
                  }
            chatInput.setText("");
                  break;
            default:
                  break;
```

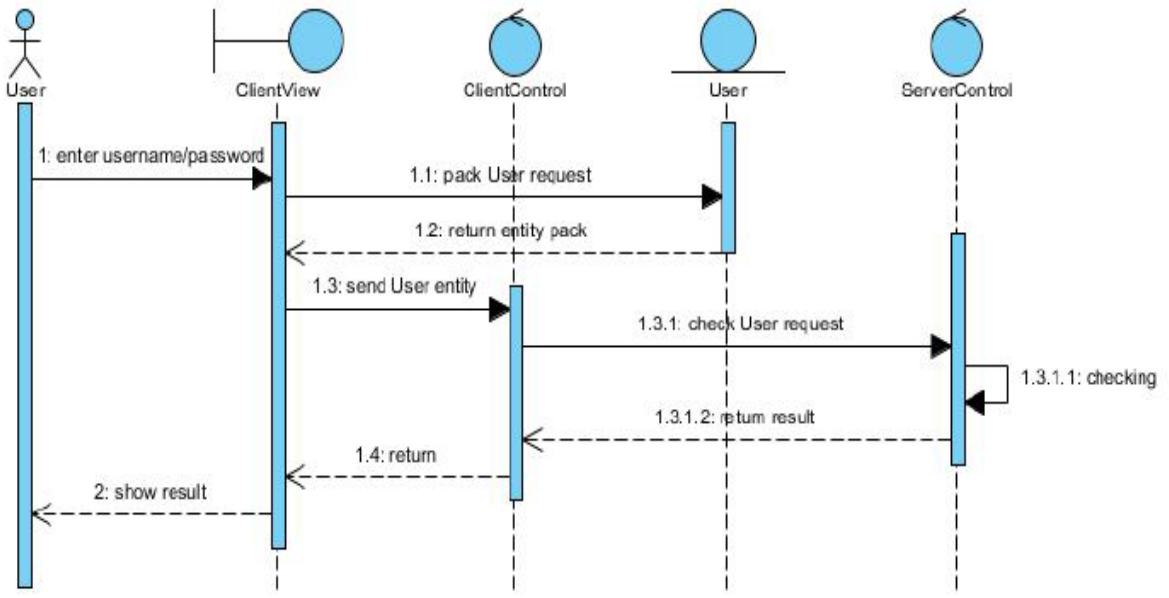
```
            }
      }
      private void sendMsg(String msg) throws IOException {
            DatagramPacket outPacket = new DatagramPacket(msg.getBytes(),
msg.length(), address, port);
            socket.send(outPacket);
      }
}
```

Bài 4. Kết hợp CSDL, xây dựng chương trình Login từ xa dùng giao thức UDP

Bài toán login từ xa dùng giao thức UDP đặt ra như sau:

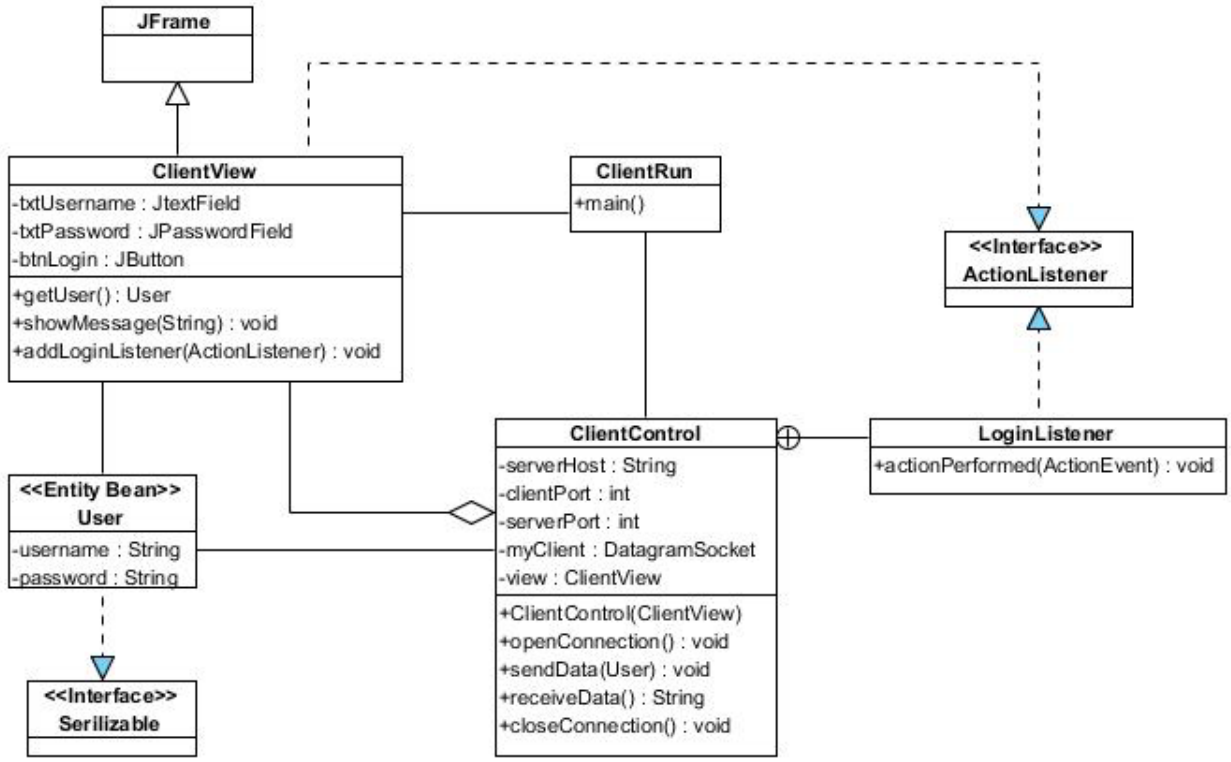
- CSDL được lưu trữ và quản lí trên Server UDP, trong đó có bảng users chứa ít nhất hai cột: cột username và cột password.
- Chương trình phía Client UDP hiện giao diện đồ họa, trong đó có một ô text để nhập username, một ô text để nhập password, và một nút nhấn Login.
- Khi nút Login được click, chương 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 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 username/password không đúng. Yêu cầu kiến trúc hệ thống được thiết kế theo mô hình MVC.

Hướng dẫn



Hình 11. Tuần tự các bước thực hiện theo giao thức UDP

Xây dựng sơ đồ lớp phía Client : được thiết kế theo mô hình MVC bao gồm 3 lớp chính tương ứng với sơ đồ M-V-C như sau:



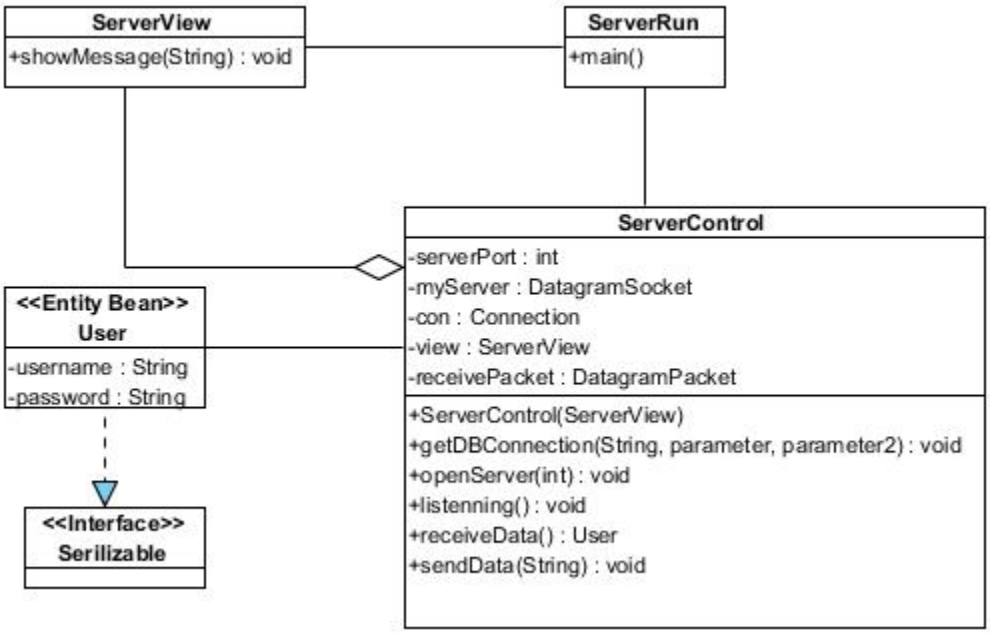
Hình 12. Sơ đồ lớp phía Client

Lớp User: 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: 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 Login.

Lớp ClientControl: tương ứng với thành phần control (C), chứa một lớp nội tại là LoginListener. Khi nút Login trên tầng view click thì nó sẽ chuyển tiếp sự kiện xuống lớp 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.



Hình 13. Sơ đồ lớp phía Server

Sơ đồ lớp của phía Server được thiết kế theo mô hình MVC, bao gồm 3 lớp chính tương ứng với sơ đồ M-V-C như sau:

Lớp User: Lớp thực thể, dùng chung thống nhất với lớp phía bên Client (M).

Lớp ServerView: 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: Tương ứng với thành phần control (C), đảm nhiệm vai trò xử lí của Server UDP, bao gồm: nhận thông tin đăng nhập từ phía các Client, kiểm tra trong CSDL xem các thông tin này đúng hay sai, sau đó gửi kết quả đăng nhập cho Client.

Sinh viên vận dụng các kiến thức đã học về lập trình mạng và kết nối CSDL tiến hành viết các lớp.

Bước 1: **Tạo các lớp phía Client**

Lớp User

```
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;
  }
```

Lớp ClientView

```
 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());
  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);
   }
});
    }
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

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

```
       }
     return result;
   }
}
```

Tạo lớp ClientRun

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

Bước 2: Các lớp phía Server

Tạo lớp ServerView

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

Tạo lớp ServerControl

```
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.getStackTrace().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.readObject();
     } catch (Exception ex) {
      view.showMessage(ex.getStackTrace().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;
      }
    }
```

Tạo lớp ServerRun

```
package udp.Server;
public class ServerRun {
```

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

Kết quả:

Login lỗi

Login thành công

TÀI LIỆU THAM KHẢO

[1]. Cay S. Horstmann, *Core Java Volum I - Fundamentals, Tenth Edition*, NewYork : Prentice Hall, 2016.

[2]. Cay S. Horstmann. *Core Java Volum II - Advanced Features, Tenth Edition*, New York : Prentice Hall, 2017.

[3].Eng.haneen Ei-masry, *Java database connection*, Islamic University of Gaza Faculty of Engineering Department of Computer Engineering ECOM 4113: DataBase Lab, 2014.

[4]. Angelos Stavrou, *Advanced Network Programming Lab using Java*, Network Security, ISA 656, Angelos Stavrou.

[5]. Marenglen Biba, Ph.D, *Manual for Lab practices, Remote Method Invocation Three Tier Application with a Database Server*, Department of Comsputer Science, University of New York.

[6].Elliotte Rusty Harold, *Java Network Programming, Fourth Edition*, O'Reilly Media, 2013.

[7]. Đoàn Văn Ban, *Lập trình hướng đối tượng với JAVA*, Nhà xuất bản Khoa học và Kỹ thuật, 2005.

[8]. ThS. Dương Thành Phết, *Bài tập thực hành Chuyên đề 1 CNPM- Java*, Khoa CNTT- Trường ĐH Công nghệ TP.HCM.

[9]. https://www.oracle.com/technetwork/java/socket-140484.html#

[10]. https://personales.unican.es/corcuerp/java/Labs/LAB_22.htm

[11]. http://www.nrcmec.org/pdf/Manuals/CSE/student/2-2%20java16-17.pdf

[12]. http://cse.mait.ac.in/pdf/LAB%20MANUAL/JAVA.pdf

[13]. https://www.academia.edu/35283541/Bài tập môn lập trình hướng đối tượng