



LẬP TRÌNH JAVA 3

BÀI 7: NETWORKING, SOCKET, SOCKET TCP, SOCKET UDP

PHẦN 1

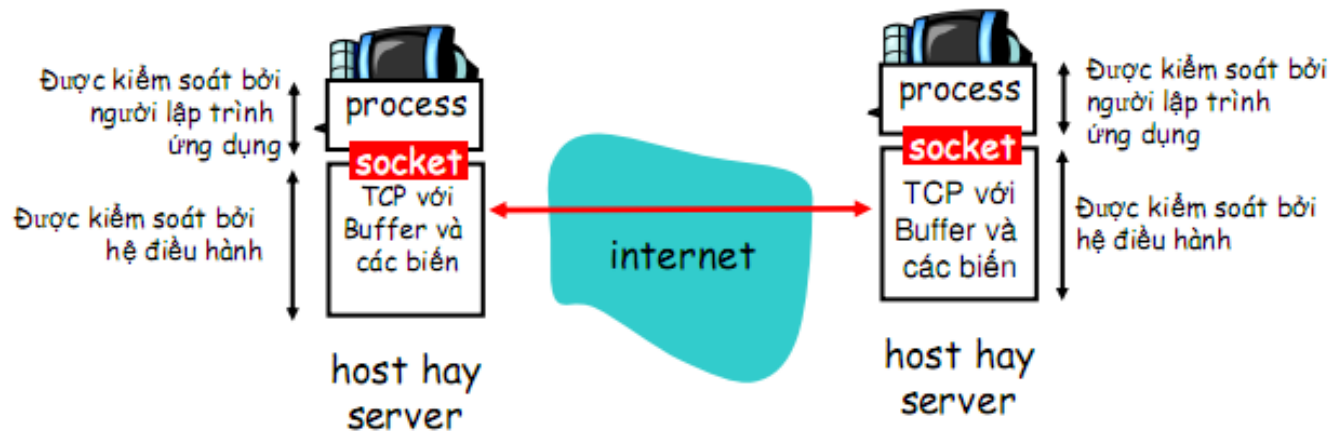
- ◎ Kết thúc bài học này bạn có khả năng
 - ❖ Giới thiệu Socket
 - ❖ Các lớp cần thiết của gói thư viện java.net
 - ❖ Lập trình Socket TCP
 - ❖ Lập trình Socket UDP
 - ❖ Bài tập



- ❑ Socket là giao diện lập trình ứng dụng (API) hay bộ thư viện hàm hỗ trợ, dùng để nối kết chương trình ứng dụng với lớp mạng trong hệ thống mạng TCP/IP.
- ❑ Được giới thiệu trong BSD4.1 UNIX, 1981
- ❑ Được khởi tạo, sử dụng và hủy một cách tường minh bởi ứng dụng
- ❑ Mô hình client/Server

❑ Cơ chế giao tiếp

- ❖ Một trong hai quá trình phải công bố số hiệu cổng của socket mà mình sử dụng để nhận và gửi dữ liệu.
- ❖ Các quá trình khác có thể giao tiếp với quá trình đã công bố cổng cũng bằng cách tạo ra một socket.



- ❑ Socket: Như là cửa thông giữa các quá trình ứng dụng và giao thức truyền tải end-to-end(UDP hay TCP)
- ❑ TCP: là dịch vụ truyền tin cậy theo **bytes** từ tiến trình này đến tiến trình khác.

❑ Hai loại dịch vụ truyền tải qua socket API:

❖ Datagram không bảo đảm (UDP)

❖ Connection-oriented bảo đảm (TCP)

❑ So sánh giữa TCP và UDP:

Có nối kết (TCP)	Không nối kết (UDP)
Tồn tại kênh giao tiếp ảo giữa 2 quá trình	Không tồn tại kênh giao tiếp ảo giữa 2 quá trình
Dữ liệu được gửi đi theo chế độ bảo đảm : có kiểm tra lỗi, truyền lại gói tin lỗi hay mất, bảo đảm thứ tự đến của các gói tin ...	Dữ liệu được gửi đi theo chế độ không bảo đảm : Không kiểm tra lỗi, không phát hiện và không truyền lại gói tin bị lỗi hay bị mất, không bảo đảm thứ tự đến của các gói tin ...
Dữ liệu chính xác Tốc độ truyền chậm	Dữ liệu không chính xác Tốc độ truyền nhanh
Thích hợp cho các ứng dụng cần độ chính xác cao: truyền file, thông tin điều khiển ...	Thích hợp cho các ứng dụng cần tốc độ, không cần chính xác cao: truyền âm thanh, hình ảnh ...

- ❑ Cổng (port): là 1 số 16 bit
 - ❖ Từ 0 – 1023: cổng hệ thống
 - ❖ Từ 1024 – 49151: cổng đã đăng ký (registered port)
 - ❖ Từ 49152 – 65535: cổng dùng riêng (private port).
- ❑ Một số cổng thông dụng
 - ❖ Echo: cổng 7 (TCP, UDP)
 - ❖ Web: cổng 80 (TCP)
 - ❖ FTP: cổng 21 cho nối kết và 20 cho dữ liệu (TCP)
 - ❖ SMTP: cổng 25 (TCP)
 - ❖ POP: cổng 110 (TCP)
 - ❖ Telnet: cổng 23 (TCP)
 - ❖ DNS: cổng 53 (TCP và UDP)
 - ❖ SNMP: cổng 161 (UDP)
 - ❖ RIP: cổng 520 (UDP)





- ☐ InetAddress
- ☐ ServerSocket
- ☐ Socket

❑ Các phương lớp của lớp InetAddress:

- ❖ `static InetAddress getLocalHost():` trả về địa chỉ máy cục bộ
- ❖ `static InetAddress getByName(String host):` nhận địa chỉ máy kiểu chuỗi, trả về đối tượng `InetAddress` thay mặt cho địa chỉ máy này.
- ❖ `public String getHostName():` trả về tên của đối tượng `InetAddress` theo dạng `String`.
- ❖ `public byte[] getAddress():` trả về địa chỉ IP của đối tượng `InetAddress` theo dạng mảng các byte.
- ❖ `public String.getHostAddress():` trả về địa chỉ IP của đối tượng `InetAddress` theo dạng `String`.

□ Ví dụ: Tìm địa chỉ IP của localhost

```
public class Demo1 {  
    public static void main(String[] args) {  
        try{  
            InetAddress myHost = InetAddress.getLocalHost();  
            System.out.println("Host address: "+myHost.getHostAddress());  
            System.out.println("Host name: "+myHost.getHostName());  
        } catch (UnknownHostException ex) {  
            ex.printStackTrace();  
        }  
    }  
}
```

Output - DemoSOF203 (run) ☒	
	run:
	Host address: 192.168.72.1
	Host name: SCD050718
	BUILD SUCCESSFUL (total time: 16 seconds)

❑ Ví dụ: Tìm các địa chỉ IP của "dantri.com.vn"

```
public class Demo2 {  
    public static void main(String[] args) {  
        try{  
            InetAddress []address = InetAddress.getAllByName("dantri.com.vn");  
            for(int i=0;i<address.length;i++){  
                System.out.println("Address "+(i+1)+" : "+address[i]);  
            }  
        }catch(UnknownHostException ex){  
            ex.printStackTrace();  
        }  
    }  
}
```

Output - DemoSOF203 (run) ☒



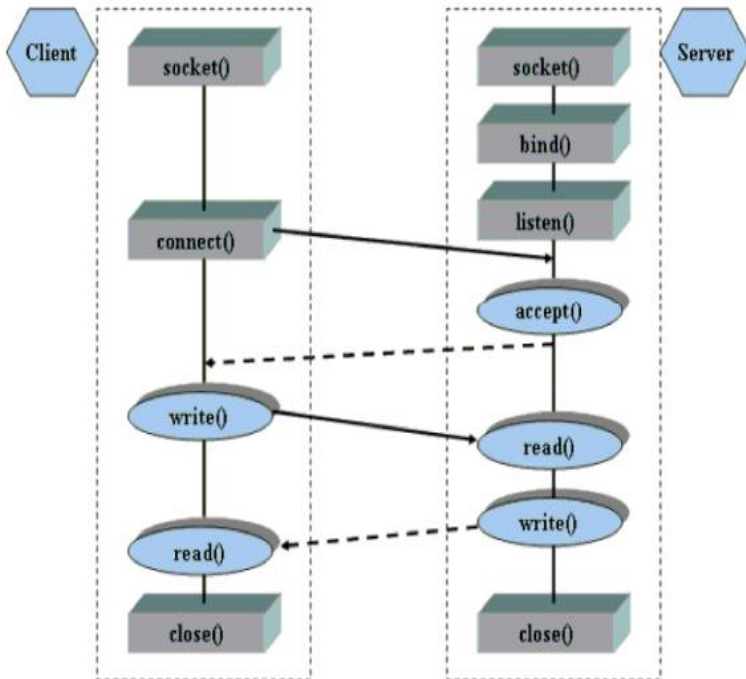
```
run:  
Address 1: dantri.com.vn/123.30.215.27  
Address 2: dantri.com.vn/123.30.215.63  
BUILD SUCCESSFUL (total time: 0 seconds)
```

❑ Các phương lớp của lớp Socket:

- ❖ `public Socket(String host, int port)`: tạo một kết nối theo địa chỉ host và số cổng port.
- ❖ `public Socket(InetAddress address, int port)`: tạo một kết nối theo địa chỉ là đối tượng `InetAddress` và số cổng port.
- ❖ `public Socket(String host, int port, boolean stream)`: tạo một kết nối theo địa chỉ host và số cổng port, `stream = true` để quy định kết nối theo TCP, ngược lại, kết nối theo UDP (User Datagram Protocol).
- ❖ `InputStream getInputStream()`: Lấy về luồng nhập để nhận dữ liệu.
- ❖ `OutputStream getOutputStream()`: Lấy về luồng xuất để gửi dữ liệu.
- ❖ `int getPort()`: Lấy về số hiệu cổng kết nối của máy chủ.
- ❖ `synchronized void close()`: Cắt đứt kết nối với máy chủ.

❑ Các phương lớp của lớp ServerSocket:

- ❖ `public ServerSocket(int port)`: tạo một đối tượng lắng nghe những kết nối từ máy khách theo số cổng port.
- ❖ `Socket accept()`: dừng lại chờ cho đến khi nhận được kết nối và trả về đối tượng Socket của máy khách.
- ❖ `synchronized void close()`: Cắt đứt kết nối với máy khách.

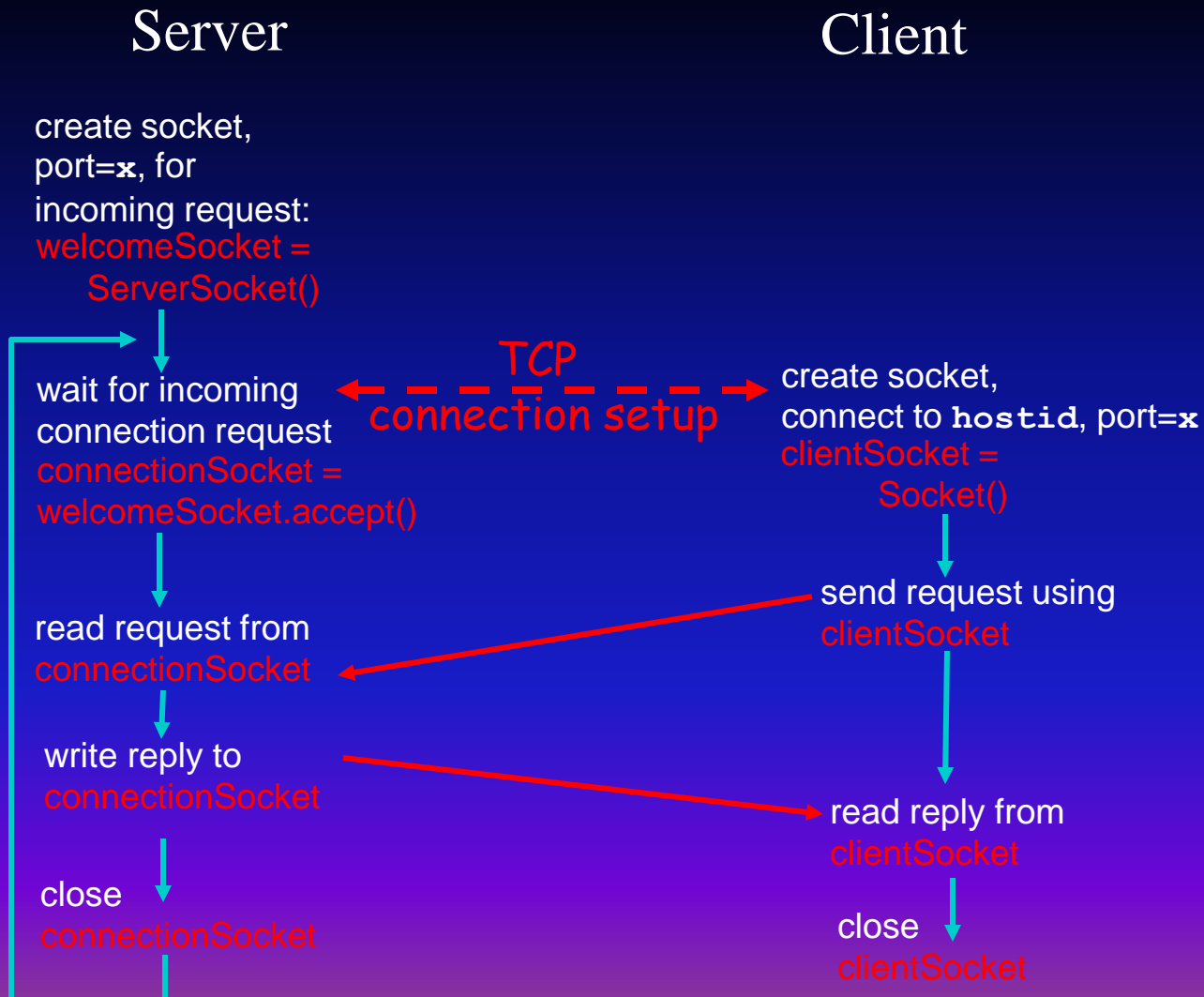


❑ Server

- ❖ Server process phải chạy trước
- ❖ Server phải tạo một socket để lắng nghe và chấp nhận các kết nối từ client

❑ Client

- ❖ Khởi tạo TCP socket
 - ❖ Xác định IP address, port của server
 - ❖ Thiết lập kết nối đến server.
- ❑ Khi server nhận yêu cầu kết nối, nó sẽ chấp nhận yêu cầu và khởi tạo socket mới để giao tiếp với client
- ❖ Server có thể chấp nhận nhiều yêu cầu client tại một thời điểm



Ví dụ: tạo ứng dụng client-server như sau

- ❑ Client đọc một dòng kí tự từ input chuẩn (**inFromUser** stream) , gửi tới server qua socket (**outToServer** stream)
- ❑ server đọc dòng kí tự trong sockets
- ❑ server biến đổi dòng kí tự đó thành dòng kí tự chỉ gồm các chữ hoa và gửi trả về cho client.
- ❑ client đọc, in ra dòng kí tự đã biến đổi từ socket (**inFromServer** stream)

❑ Client

```
import java.io.*;  
import java.net.*;  
class TCPCClient {
```

```
    public static void main(String argv[]) throws Exception  
    {
```

```
        String sentence;  
        String modifiedSentence;
```

Tạo input stream



```
        BufferedReader inFromUser =  
            new BufferedReader(new InputStreamReader(System.in));
```

Tạo client socket,
kết nối tới server



```
        Socket clientSocket = new Socket("hostname", 6789);
```

Tạo output stream,
đính kèm vào socket



```
        DataOutputStream outToServer =  
            new DataOutputStream(clientSocket.getOutputStream());
```


❑ Client

Tạo input stream,
đính kèm vào trong
socket

```
BufferedReader inFromServer =  
    new BufferedReader(new  
        InputStreamReader(clientSocket.getInputStream()));
```

Gửi dòng kí tự
đến server

```
sentence = inFromUser.readLine();  
outToServer.writeBytes(sentence + '\n');
```

Đọc dòng kí tự
(đã biến đổi) gửi
về từ server

```
modifiedSentence = inFromServer.readLine();  
System.out.println("FROM SERVER: " + modifiedSentence);  
clientSocket.close();  
  
}  
}
```

❑ Server

```
import java.io.*;  
import java.net.*;
```

```
class TCPServer {
```

```
    public static void main(String argv[]) throws Exception  
    {
```

```
        String clientSentence;  
        String capitalizedSentence;
```

Tạo sẵn Socket
ở cổng 6789

```
        ServerSocket welcomeSocket = new ServerSocket(6789);
```

```
        while(true) {
```

Đợi đến khi có socket
từ client gửi đến

```
            Socket connectionSocket = welcomeSocket.accept();
```

Tạo input stream,
đính kèm vào socket

```
            BufferedReader inFromClient =  
                new BufferedReader(new  
                    InputStreamReader(connectionSocket.getInputStream()));
```

❑ Server

Tạo output stream,
đính kèm vào
socket

```
DataOutputStream outToClient =  
    new DataOutputStream(connectionSocket.getOutputStream());
```

Đọc dòng kí tự
trong socket

```
clientSentence = inFromClient.readLine();
```

```
capitalizedSentence = clientSentence.toUpperCase() + '\n';
```

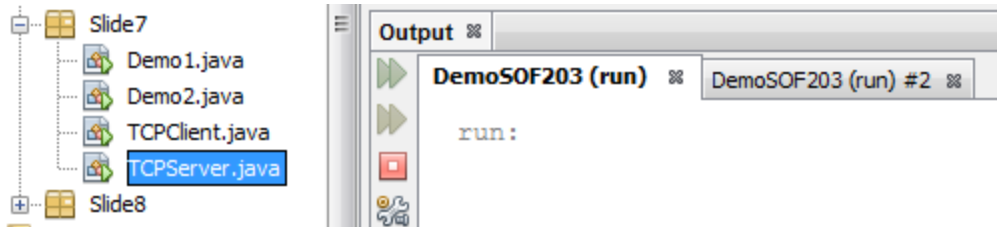
Ghi dòng kí tự đã
biến đổi vào socket

```
outToClient.writeBytes(capitalizedSentence);
```

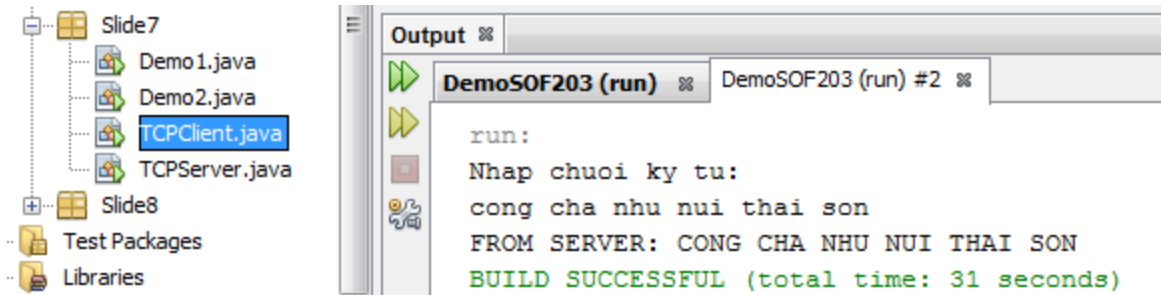
```
}  
}  
}
```

Kết thúc vòng lặp while,
quay trở về vòng lặp cha,
đợi kết nối khác

❑ Chạy bên server trước (TCPServer)



❑ Chạy bên client sau (TCPClient)





DEMO

Chạy và giải thích



- ❑ UDP không thiết lập kết nối giữa client và server
- ❑ Không “bắt tay”.
- ❑ Bên gửi phải xác định chính xác địa chỉ IP và cổng của bên nhận.
- ❑ Server xác định địa chỉ IP và cổng của bên gửi từ datagram nhận được.
- ❑ UDP cung cấp dịch vụ truyền dữ liệu không tin cậy theo byte (“datagrams”) giữa Client và Server

Server

create socket,
port=**x**, for
incoming request:
serverSocket =
DatagramSocket()

read request from
serverSocket

write reply to
serverSocket
specifying client
host address,
port umber

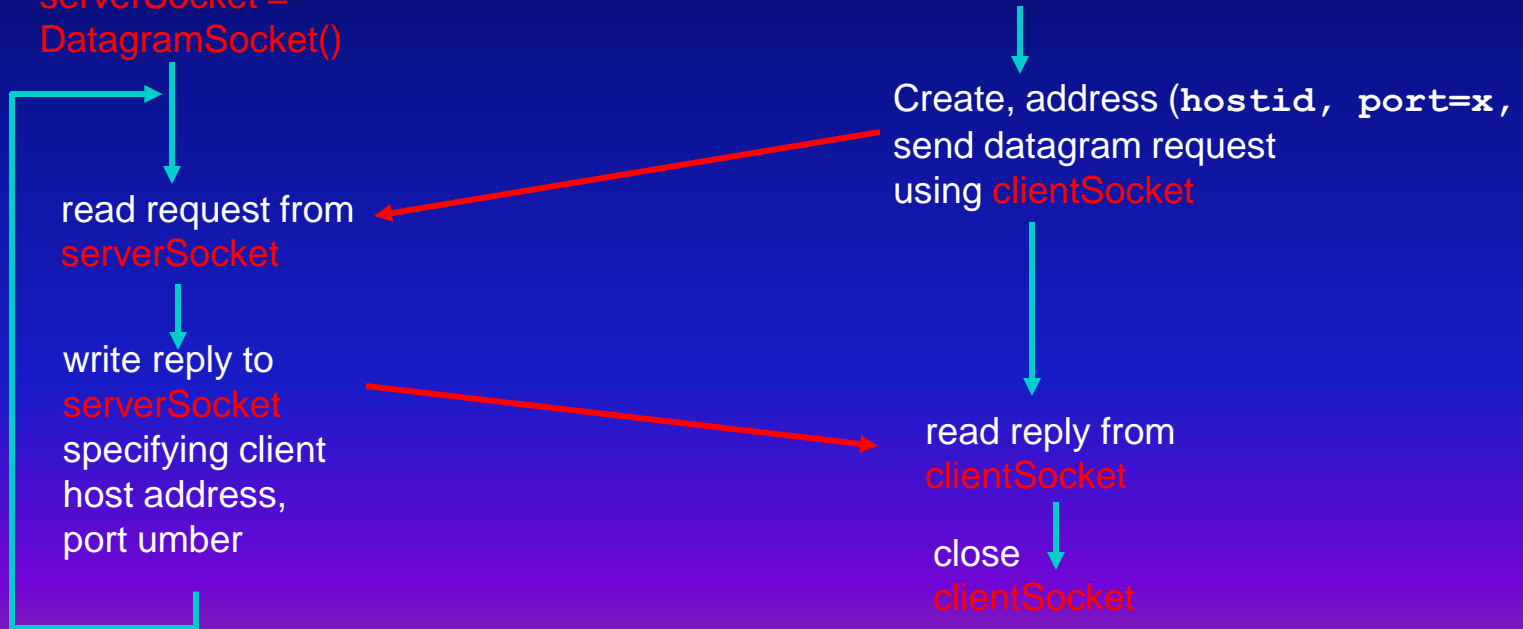
Client

create socket,
clientSocket =
DatagramSocket()

Create, address (**hostid**, **port=x**,
send datagram request
using **clientSocket**

read reply from
clientSocket

close
clientSocket



□ Client

```
import java.io.*;  
import java.net.*;
```

```
class UDPClient {  
    public static void main(String args[]) throws Exception  
    {
```

Tạo input stream



```
        BufferedReader inFromUser =  
            new BufferedReader(new InputStreamReader(System.in));
```

Tạo client socket



```
        DatagramSocket clientSocket = new DatagramSocket();
```

Chuyển hostname
thành địa chỉ IP
sử dụng DNS



```
        InetAddress IPAddress = InetAddress.getByName("hostname");
```

```
        byte[] sendData = new byte[1024];  
        byte[] receiveData = new byte[1024];
```

```
        String sentence = inFromUser.readLine();  
        sendData = sentence.getBytes();
```


❑ Client

Tạo datagram cùng
với dữ liệu, độ dài,
địa chỉ IP, cổng

```
DatagramPacket sendPacket =  
    new DatagramPacket(sendData, sendData.length, IPAddress, 6789);
```

Gửi datagram
tới server

```
clientSocket.send(sendPacket);
```

Đọc datagram
gửi về từ server

```
DatagramPacket receivePacket =  
    new DatagramPacket(receiveData, receiveData.length);
```

```
clientSocket.receive(receivePacket);
```

```
String modifiedSentence =  
    new String(receivePacket.getData());
```

```
System.out.println("FROM SERVER:" + modifiedSentence);  
clientSocket.close();  
}
```

```
}
```

❑ Server

```
import java.io.*;  
import java.net.*;
```

```
class UDPServer {  
    public static void main(String args[]) throws Exception  
    {
```

Tạo datagram socket
ở cổng 9876

```
        DatagramSocket serverSocket = new DatagramSocket(9876);
```

```
        byte[] receiveData = new byte[1024];  
        byte[] sendData = new byte[1024];
```

```
        while(true)  
        {
```

Tạo datagram nhận

```
            DatagramPacket receivePacket =  
                new DatagramPacket(receiveData, receiveData.length);
```

Nhận
datagram

```
            serverSocket.receive(receivePacket);
```

❑ Server

```
String sentence = new String(receivePacket.getData());
```

Nhận địa chỉ IP,
cổng của bên gửi

```
InetAddress IPAddress = receivePacket.getAddress();
```

```
int port = receivePacket.getPort();
```

```
String capitalizedSentence = sentence.toUpperCase();
```

```
sendData = capitalizedSentence.getBytes();
```

Tạo datagram để
gửi tới client

```
DatagramPacket sendPacket =  
    new DatagramPacket(sendData, sendData.length, IPAddress,  
        port);
```

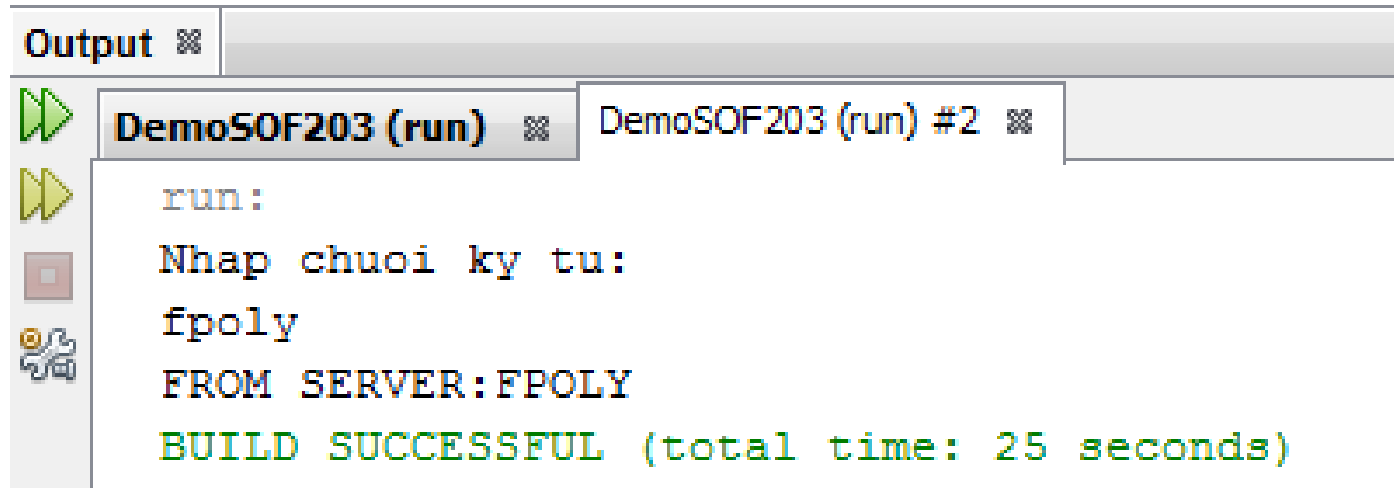
Đính datagram
vào socket

```
serverSocket.send(sendPacket);
```

```
}  
}  
}
```

Kết thúc vòng lặp while,
Quay trở về vòng lặp cha,
đợi datagram khác đến

- ❑ Chạy file UDPServer trước, file UDPClient sau



```
Output
DemoSOF203 (run) DemoSOF203 (run) #2
run:
Nhap chuoi ky tu:
fpoly
FROM SERVER:FPOLY
BUILD SUCCESSFUL (total time: 25 seconds)
```



DEMO

Chạy và giải thích



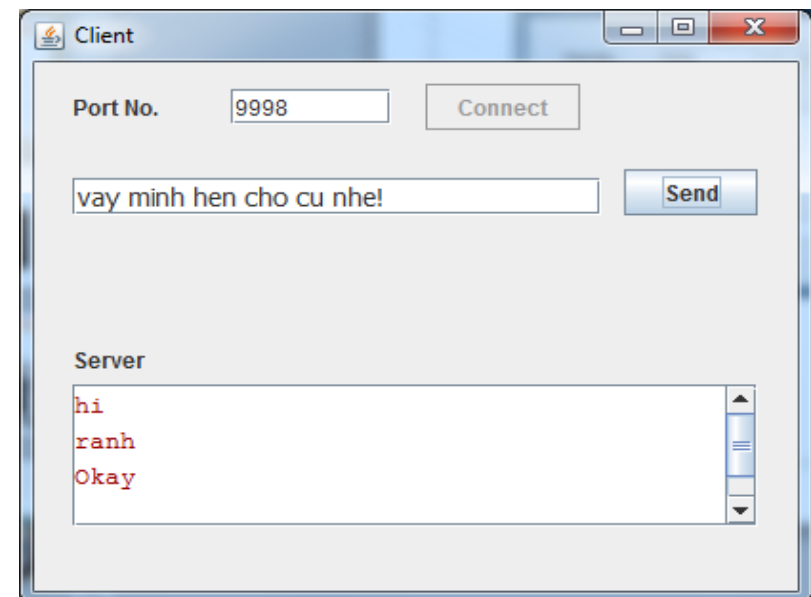
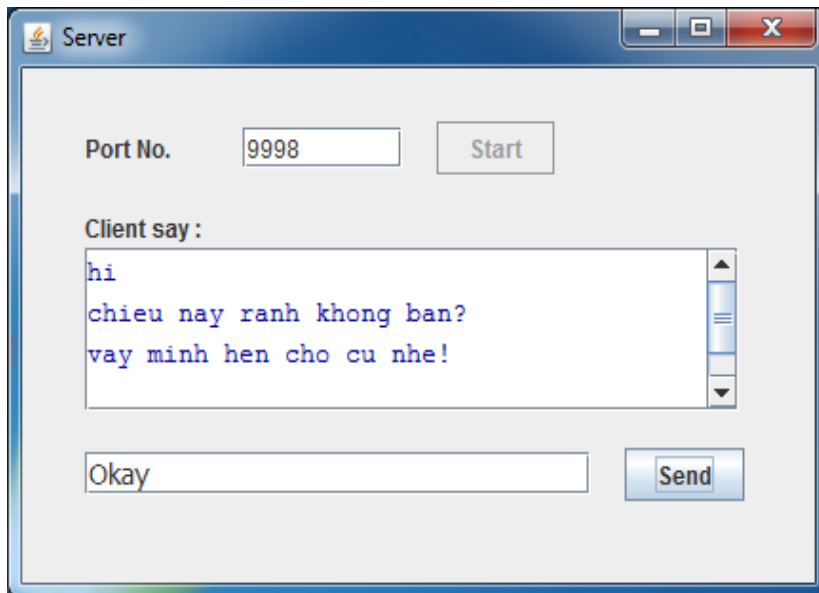


LẬP TRÌNH JAVA 3

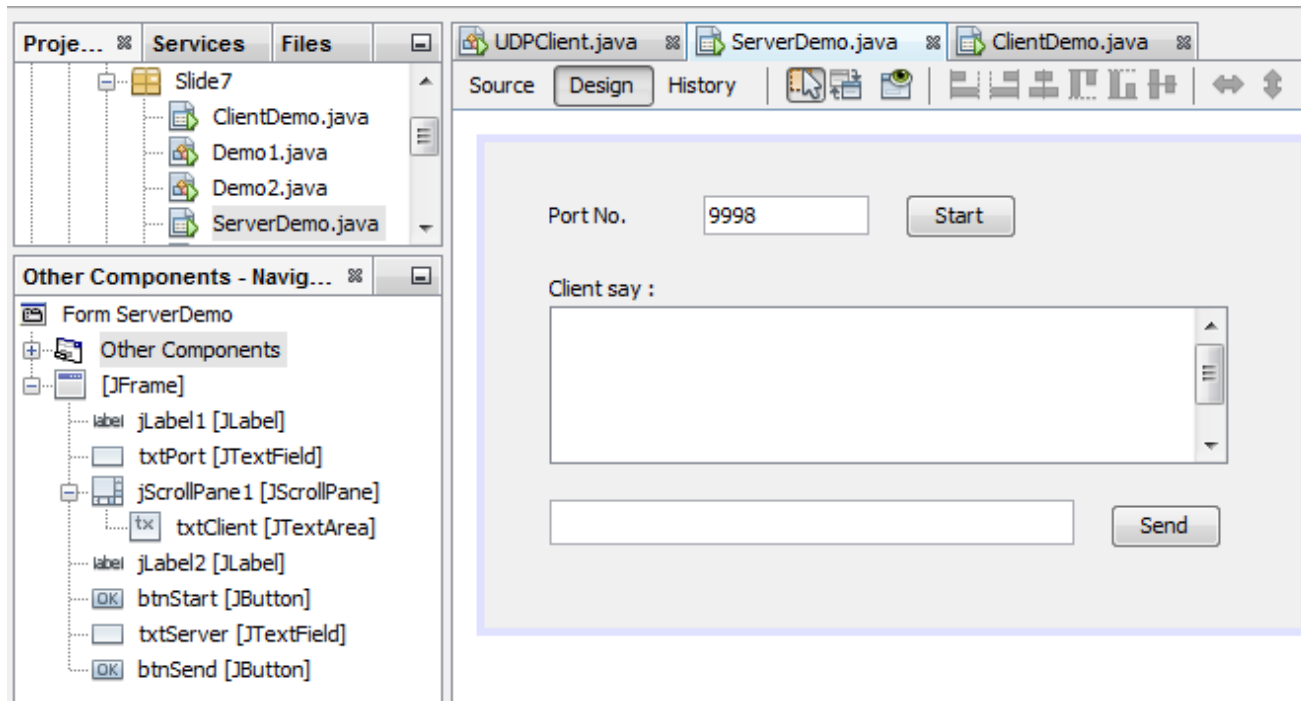
BÀI 7: NETWORKING, SOCKET, SOCKET TCP, SOCKET UDP

PHẦN 2

❑ Viết chương trình chat



❑ Thiết kế giao diện cho Server



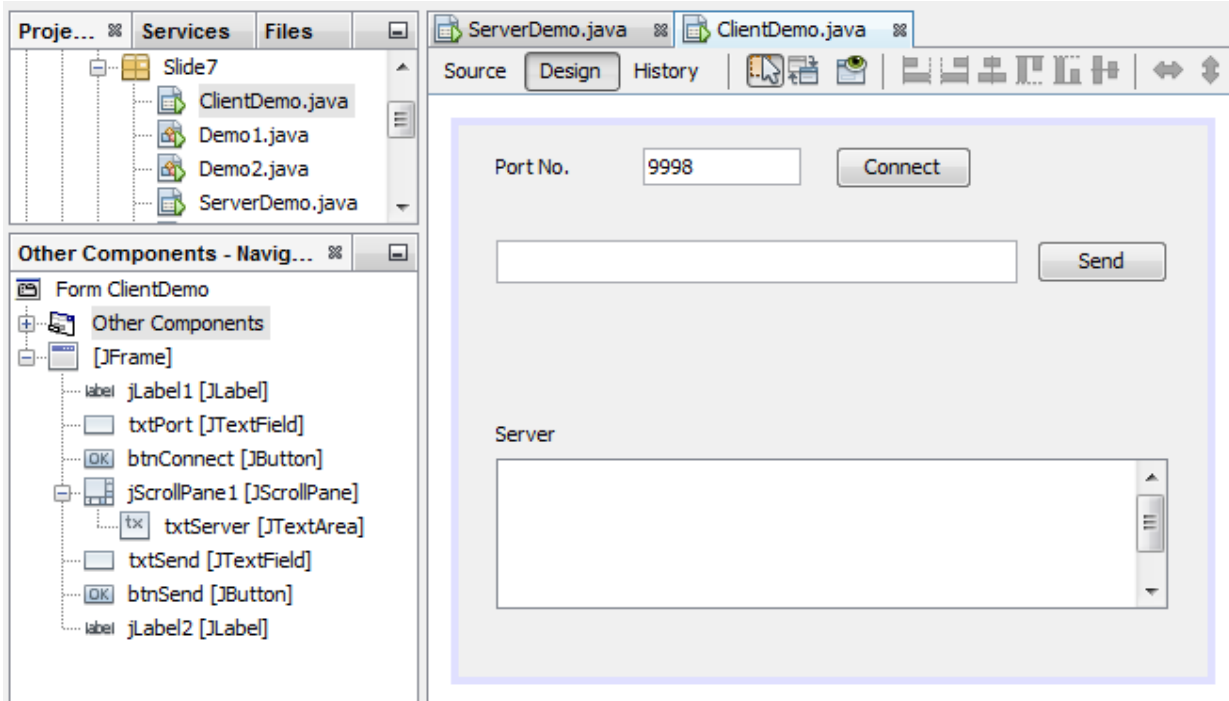
```
public class ServerDemo extends javax.swing.JFrame {
    ServerSocket server=null;
    Socket client = null;
    OutputStream out;
    PrintStream ps;
    int port;
    public ServerDemo() {
        initComponents();
        btnSend.setEnabled(false);
    }
    //.....
}
```


□ Viết code cho button Start, Send

```
private void btnStartActionPerformed(java.awt.event.ActionEvent evt) {  
    try {  
        port = Integer.parseInt(txtPort.getText().trim());  
        server = new ServerSocket(port);  
  
        client = server.accept();  
        btnStart.setEnabled(false);  
        btnSend.setEnabled(true);  
  
        out = client.getOutputStream();  
        ps = new PrintStream(out);  
        Thread t = new Thread(new ServerThread(client, txtClient));  
        t.start();  
    } catch (IOException ex) {  
        ex.printStackTrace();  
    }  
}
```

```
private void btnSendActionPerformed(java.awt.event.ActionEvent evt) {  
    String s = txtServer.getText().trim();  
    ps.println(s);  
}
```

Thi t k  giao di n cho Client



```
* @author Tu Ech
*/
public class ClientDemo extends javax.swing.JFrame {
    int port;
    Socket client;
    OutputStream out;
    PrintStream ps;
    public ClientDemo() {
        initComponents();
        btnSend.setEnabled(false);
    }
    //.....
```

□ Viết code cho button Connect, Send

```
private void btnConnectActionPerformed(java.awt.event.ActionEvent evt) {  
    try {  
        port = Integer.parseInt(txtPort.getText().trim());  
        client = new Socket("127.0.0.1", port);  
        out = client.getOutputStream();  
        ps = new PrintStream(out);  
  
        btnConnect.setEnabled(false);  
        btnSend.setEnabled(true);  
        Thread t = new Thread(new ClientThread(client, txtServer));  
        t.start();  
  
    } catch (Exception ex) {  
        ex.printStackTrace();  
    }  
}
```

```
private void btnSendActionPerformed(java.awt.event.ActionEvent evt) {  
    ps.println(txtSend.getText().trim());  
}
```



DEMO

Chạy và giải thích



- ❖ Giới thiệu Socket
- ❖ Các lớp cần thiết của gói thư viện java.net
- ❖ Lập trình Socket TCP
- ❖ Lập trình Socket UDP
- ❖ Bài tập





Cảm ơn