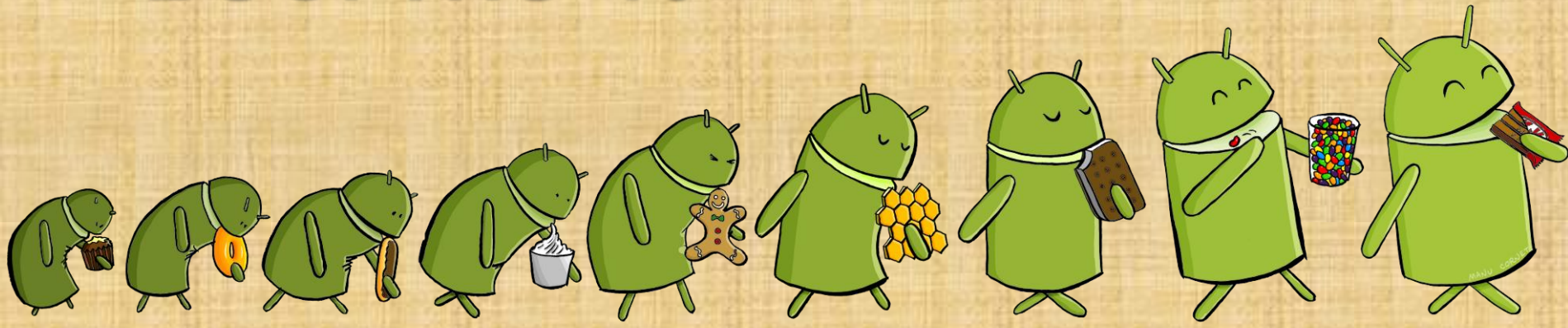


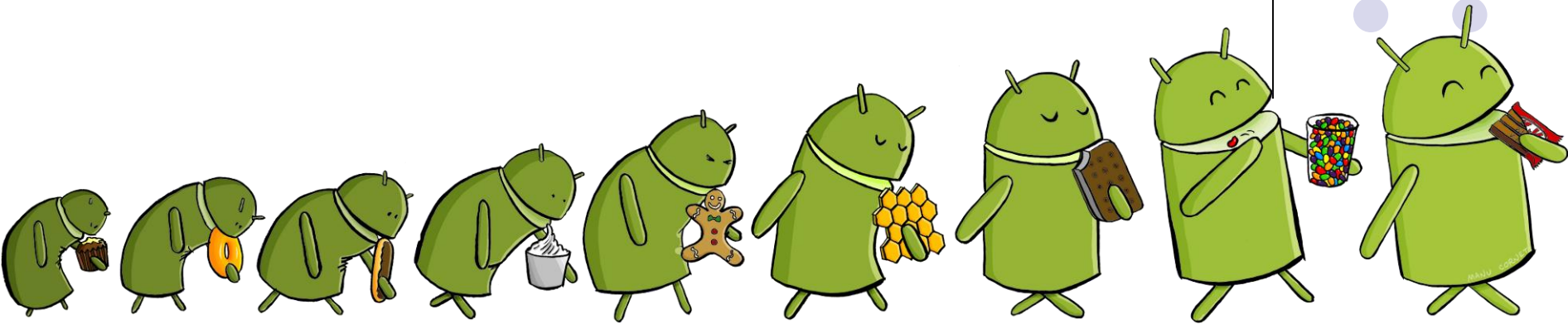
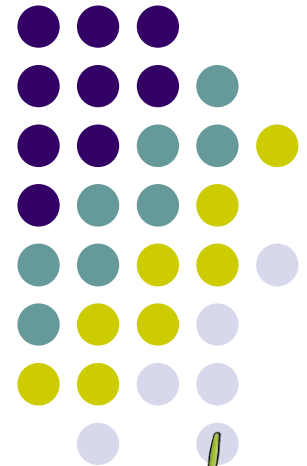


LỚP CHUYÊN ĐỀ LẬP TRÌNH DI ĐỘNG ANDROID
KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT
BUỔI THỨ 10



LẬP TRÌNH SOCKET TRÊN ANDROID

Nguyễn Minh Đạo
Nguyễn Đỗ Anh Khoa





NỘI DUNG

- Khái niệm về Socket.
- Hai loại socket: UDP và TCP.
- Lập trình UDP.
- Lập trình TCP.
- Bài tập áp dụng.

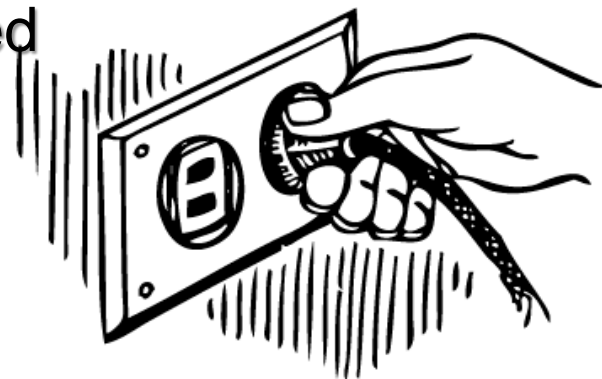


KHÁI NIỆM VỀ SOCKET



- Socket API

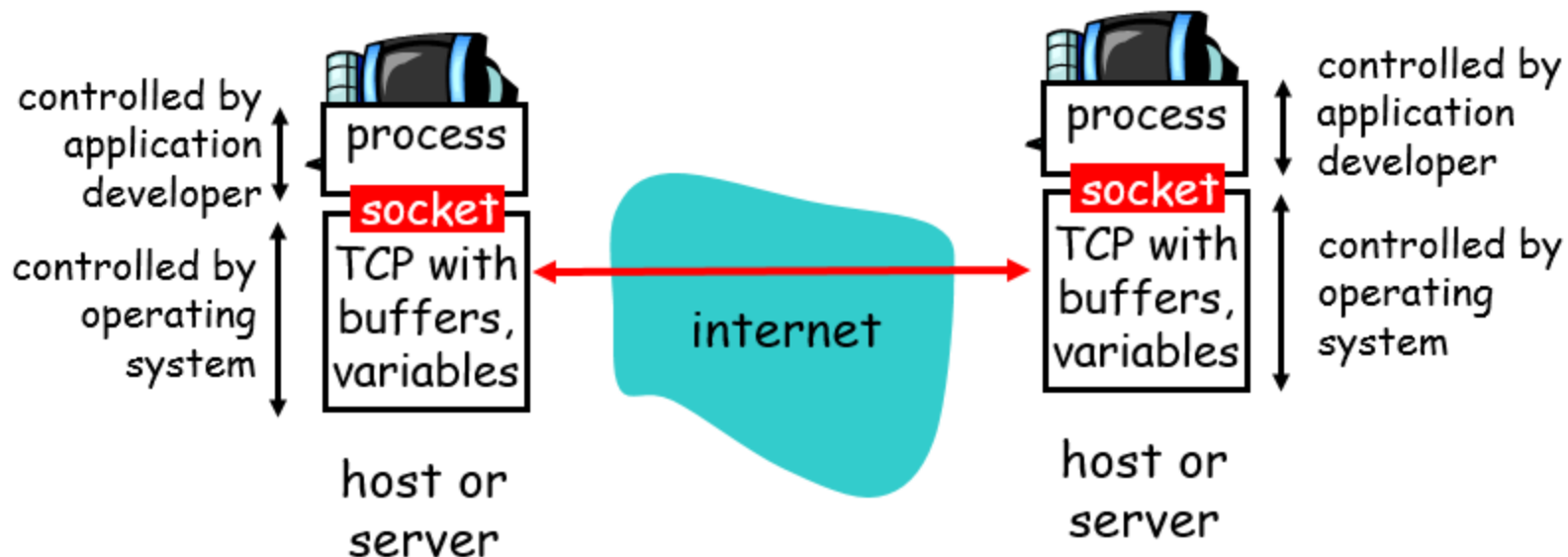
- Được giới thiệu ở BSD4.1 UNIX, 1981
- Được ứng dụng khởi tạo, sử dụng và hủy bỏ.
- Dùng cơ chế client/server
- Cung cấp hai dịch vụ chuyển dữ liệu thông qua socket API:
 - unreliable datagram
 - reliable, byte stream-oriented



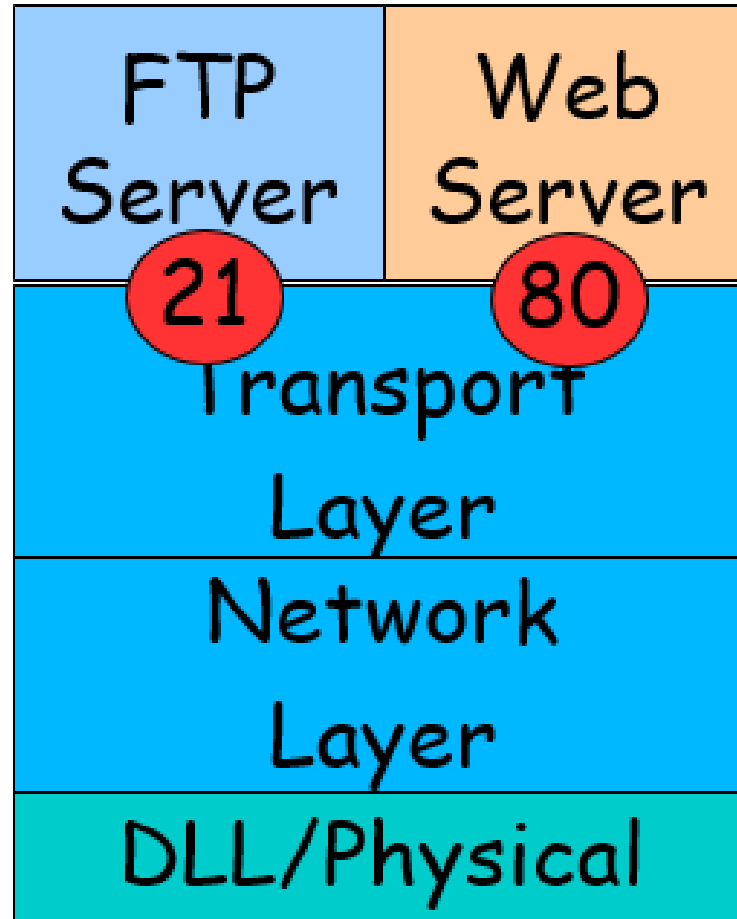
KHÁI NIỆM VỀ SOCKET



- **Socket**: “cửa” nằm giữa process ứng dụng và end-end transport protocol (UDP và TCP).
- **TCP service**: dịch vụ truyền tin cậy chuỗi bytes giữa hai process.



Port Number



2 loại Socket



- Stream Socket:
 - Dựa trên giao thức TCP.
- Datagram Socket:
 - Dựa trên giao thức UDP.



UDP(User Datagram Protocol)



- Không yêu cầu kết nối.
- Đơn giản, nhanh, nhưng không tin cậy.
- Dữ liệu được truyền nhận theo từng gói tin đơn lẻ.
- Khi xảy ra lỗi phải thực hiện truyền lại toàn bộ gói tin.



TCP(Transmission Control Protocol)



- Hướng kết nối
- Đảm bảo độ tin cậy trong quá trình truyền/nhận dữ liệu.
- Dữ liệu được chia thành các gói tin và được đánh số thứ tự.
- Từng gói tin sẽ được xác nhận truyền/nhận thành công.





Socket Programming





Lập trình UDP

- Sử dụng 2 class:
 - **DatagramSocket**
 - **DatagramPacket**
- **DatagramPacket** đóng gói các byte dữ liệu vào các gói tin UDP được gọi là datagram và cho phép ta mở các datagram khi nhận được.
- Một **DatagramSocket** đồng thời thực hiện cả hai nhiệm vụ nhận và gửi gói tin.





Lập trình UDP

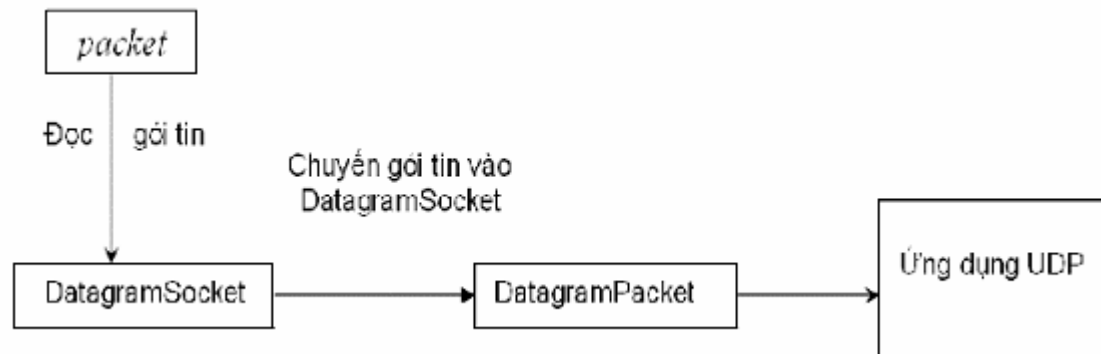
- Gửi dữ liệu:
 - dữ liệu → **DatagramPacket** → gửi đi thông qua **DatagramSocket**.
- Nhận dữ liệu:
 - nhận **DatagramPacket** thông qua **DatagramSocket** → đọc nội dung **DatagramPacket**





Lập trình UDP

- Để gửi dữ liệu, ta đặt dữ liệu trong một DatagramPacket và gửi gói tin bằng cách sử dụng DatagramSocket.
- Để nhận dữ liệu, ta nhận một đối tượng DatagramPacket từ DatagramSocket và sau đó đọc nội dung của gói tin.



Lập trình UDP



Server

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

read request from
serverSocket

write reply to
serverSocket
specifying client
host address,
port number

Client

create socket,
**clientSocket =
DatagramSocket()**

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

read reply from
clientSocket

close
clientSocket



Thí dụ 1: Kết Nối UDP Sử Dụng Socket, chuẩn gửi nhận SỐ - SỐ



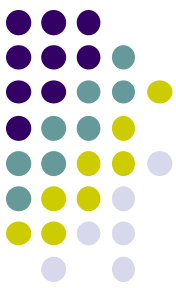
```

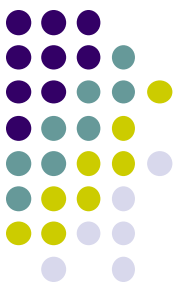
public class UDPSS
{
    public static void main(String[] args) throws Exception
    {
        int port = 2812;
        DatagramSocket socket = new DatagramSocket(port);
        DatagramPacket packet;
        byte []data;
        try{
            data = new byte[1024];
            packet = new DatagramPacket(data,data.length);
            socket.receive(packet);
            String st = new String(packet.getData(),0,packet.getLength());
            double so = Double.parseDouble(st);
            InetAddress ipC = packet.getAddress();
            int portC = packet.getPort();

            double kqD = XuLy(so);
            String kqS = String.valueOf(kqD);
            data = kqS.getBytes();
            packet = new DatagramPacket(data,data.length,ipC,portC);
            socket.send(packet);
            socket.close();
        }
        catch (UnknownHostException evt){ evt.printStackTrace(); }
    }
    public static double XuLy(double so)
    {
        return Math.sqrt(so);
    }
}

```

SERVER





```
class UDPCS
{
    public static void main(String[] args) throws Exception
    {
        DatagramSocket socket = new DatagramSocket();
        DatagramPacket packet;
        byte []data;
        InetAddress ipS = InetAddress.getByName("localhost");
        int ports = 2812;
        BufferedReader in = new BufferedReader(new
            InputStreamReader(System.in));
        System.out.print("Nhap so bat ky : ");
        double so = Double.parseDouble(in.readLine());
        String st = String.valueOf(so);
        data = st.getBytes();
        packet = new DatagramPacket(data,data.length,ipS,ports);
        socket.send(packet);
        data = new byte[1024];
        packet = new DatagramPacket(data,data.length);
        socket.receive(packet);
        String kqS = new String(packet.getData(),0,packet.getLength());
        double kqD = Double.parseDouble(kqS);
        System.out.println("Can Bac Hai Tinh Duoc : " + kqD);
        socket.close();
    }
}
```

CLIENT

Kết quả

A screenshot of an IDE console window. The title bar shows tabs for 'Problems', '@ Javadoc', 'Declaration', 'Console', and 'LogCat'. The console text reads: '<terminated> UDPCS [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (Jan 14, 2014 11:24:26 PM)' followed by two lines of output: 'Nhap so bat ky : 1234' and 'Can Bac Hai Tinh Duoc : 35.12833614050059'. A scrollbar is visible at the bottom of the console area.

```
<terminated> UDPCS [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (Jan 14, 2014 11:24:26 PM)
Nhap so bat ky : 1234
Can Bac Hai Tinh Duoc : 35.12833614050059
```





Lập trình TCP

- Server
 - 2 class: **ServerSocket**, **Socket**
 - **ServerSocket** lắng nghe yêu cầu kết nối từ phía Client.
 - Khi có yêu cầu kết nối, server sẽ tạo ra 1 **Socket** để giao tiếp với Client, và tiếp tục chờ yêu cầu kết nối từ một client khác.



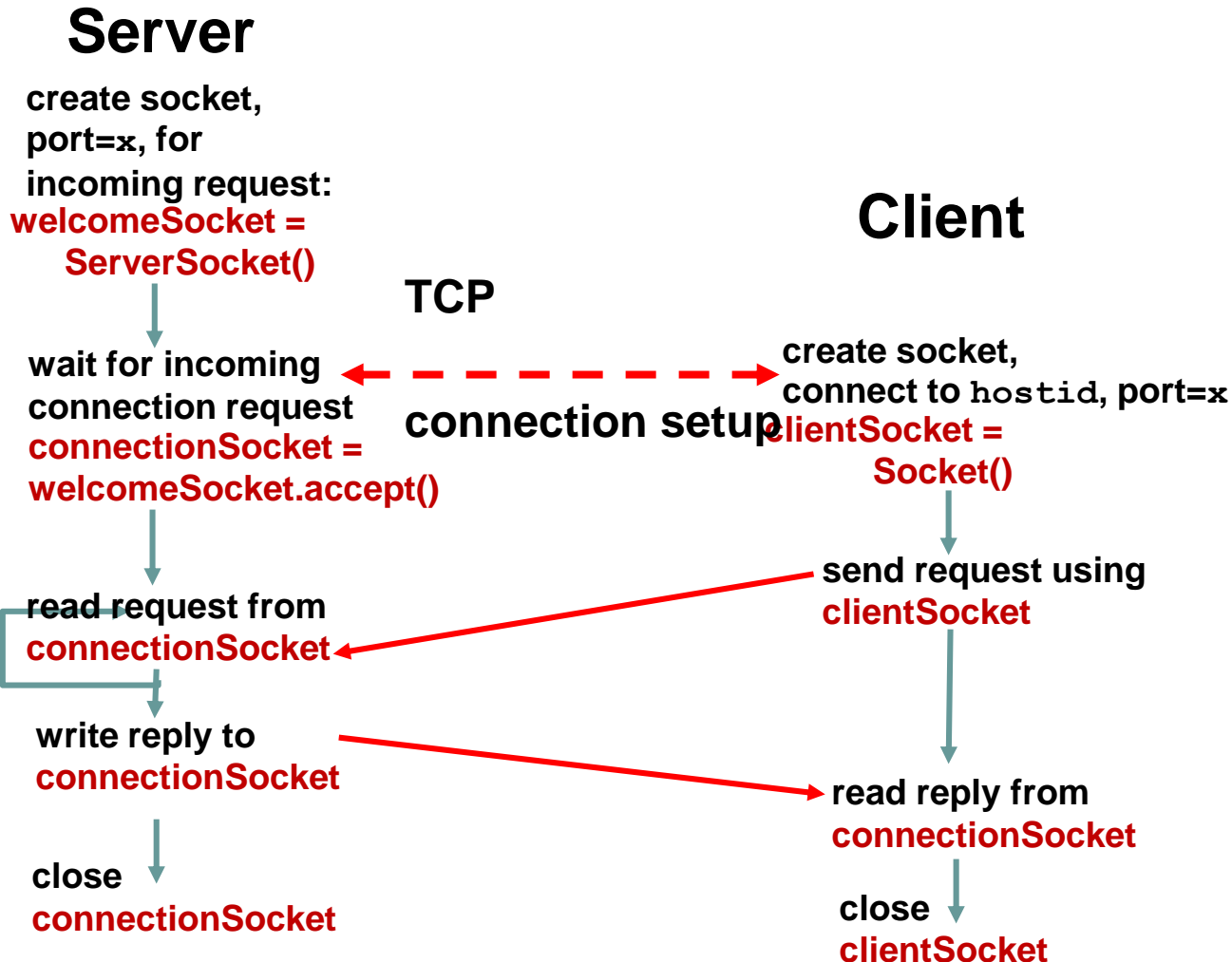


Lập trình TCP

- Client
 - Class: Socket
 - Tạo 1 Socket, chỉ định rõ địa chỉ IP và Port của Server
 - Khi Socket được tạo thành công, thì một kết nối đến Server được thiết lập.



TCP Programming



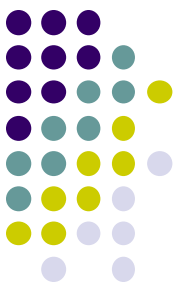


Thí dụ 1 : Socket – TCP/IP

- Phía Server:

```
Server.java Client.java TCPThread.java
package TCP;
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;
public class Server {
    @SuppressWarnings("resource")
    public static void main(String[] args) throws IOException,
        ClassNotFoundException {
        ServerSocket wellcomSocket;
        Socket connectionSocket;
        wellcomSocket=new ServerSocket(5000);
        System.out.println("Server is running...");
        while(true) {
            connectionSocket=wellcomSocket.accept();
            new TCPThread(connectionSocket);
        }
    }
}
```





Thí dụ 1 : Socket – TCP/IP

- Khởi chạy Server:



- Server lắng nghe tại port=5000
- Sau khi xuất hiện dòng thông báo, sẽ bước vào một vòng lặp chờ yêu cầu kết nối đến từ client.
- Mỗi kết nối được thiết lập sẽ do một tiểu trình quản lý (thread).

```
while(true) {  
    connectionSocket=wellcomSocket.accept();  
    new TCPThread(connectionSocket);  
}
```



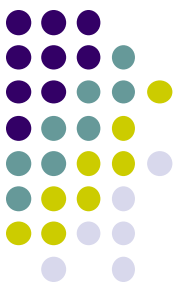
Thí dụ 1 : Socket – TCP/IP



- Chương trình tạo Thread quản lý kết nối từ Client đến Server:




```
package TCP;
import java.io.IOException;
public class TCPThread extends Thread {
    Socket connectionSocket;
    public TCPThread(Socket s){
        connectionSocket=s;
        start();
    }
    public void run(){
        ObjectInputStream is = null;
        ObjectOutputStream os=null;
        String s = null;
        while(true){
            try {
                is=new ObjectInputStream(connectionSocket.getInputStream());
                os=new ObjectOutputStream(connectionSocket.getOutputStream());
            } catch (IOException e) {
                e.printStackTrace();
            }
            try {
                s=(String)is.readObject();
            } catch (ClassNotFoundException | IOException e) {
                e.printStackTrace();
            }
            System.out.println(s);
            s=s.toUpperCase();
            try {
                os.writeObject(s);
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
}
```



Thí dụ 1 : Socket – TCP/IP

- Tập tin Client:

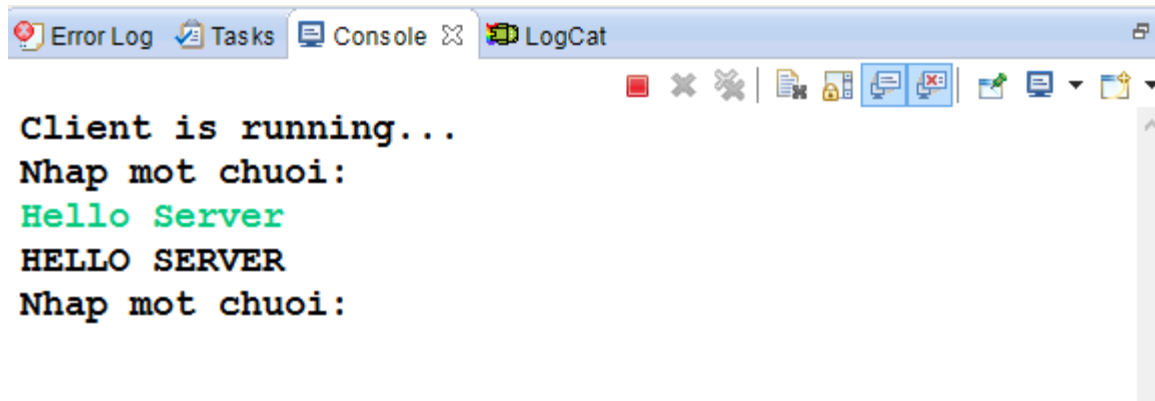
```
public class Client {  
    @SuppressWarnings("resource")  
    public static void main(String[] args) throws UnknownHostException,  
        IOException, ClassNotFoundException {  
        Socket clietnSocket;  
        ObjectOutputStream os;  
        ObjectInputStream is;  
        BufferedReader br;  
        clietnSocket=new Socket(InetAddress.getLocalHost(), 5000);  
        System.out.println("Client is running...");  
        while(true) {  
            System.out.println("Nhap mot chuoi:");  
            br=new BufferedReader(new InputStreamReader(System.in));  
            String s=br.readLine();  
            os=new ObjectOutputStream(clietnSocket.getOutputStream());  
            os.writeObject(s);  
            is=new ObjectInputStream(clietnSocket.getInputStream());  
            String t=(String)is.readObject();  
            System.out.println(t);  
        }  
    }  
}
```





Thí dụ 1 : Socket – TCP/IP

- Khởi chạy Client:



```
Error Log Tasks Console LogCat
Client is running...
Nhap mot chuoai:
Hello Server
HELLO SERVER
Nhap mot chuoai:
```

- Khi nhập một chuỗi, nội dung sẽ chuyển đến phía Server, sẽ được Server đổi nội dung sang chữ HOA, và gửi lại phía Client.



BÀI TẬP ÁP DỤNG

- Viết chương trình Chat Socket

