TRƯỜNG ĐẠI HỌC BÁCH KHOA

**KHOA Công Nghệ Thông Tin**

BỘ MÔN: MẠNG & TT

# ĐỀ THI VÀ BÀI LÀM - Đề 1

Tên học phần: Lập Trình Mạng

Mã học phần: Hình thức thi: *Tự luận có giám sát*

Thời gian làm bài: 60 phút *(không kể thời gian phát đề và nộp bài)*

Được sử dụng tài liệu khi làm bài. Không chia sẻ bài cho nhau, nếu phát hiện sẽ chia đều số điểm.

**Họ tên:**…Nguyễn Hữu Phúc …………**Lớp**:………21TCLC\_DT3………**MSSV**:……102210225……...

Sinh viên làm bài trực tiếp trên tệp này, lưu tệp với định dạng MSSV\_HọTên.pdf và nộp bài thông qua MSTeam:

***Câu 1*** (*3 điểm*): Hãy viết chương trình theo giao thức TCP với các chức năng sau:

1. Server:
   1. Lắng nghe và chấp nhận kết nối từ các client. Ứng với mỗi kết nối tạo 1 luồng xử lý sau:
      1. Chọn ngẫu nhiên 1 chuỗi **x** đúng 30 các ký tự alphabet, ví dụ:

**x**=“PhamMinhTuanPhamMinhTuanPhamMi”

* + 1. Nhận 1 chuỗi từ client gửi tới.
    2. Nếu chuỗi **y** từ client gửi tới **chứa các ký tự khác alphabet** thì thì trả về cho client chuỗi “**Khong dung dinh dang**”.
    3. Nếu chuỗi **y** đúng định dạng thì so sánh chuỗi đó với **x**.
       - Nếu **y** lớn hơn **x** theo thứ tự từ điển thì trả về cho client ký tự “**>**”, nếu nhỏ hơn thì trả về ký tự “**<**” và tiến tới bước (v). Ví dụ: **y** = “xxx” thì phản hồi “>”, **y** = “aaaaaa” thì phản hồi “<”
       - Nếu **y** bằng **x** theo thứ tự từ điển thì trả về “**Chuc mung vi da doan dung**” và nhảy tới bước (vi).
    4. Quay về bước (ii).
    5. Đóng kết nối.

1. Client:
   1. Kết nối tới server
   2. Lặp đi lặp lại các bước sau:
      1. Nhập từ bàn phím một chuỗi alphabet. Nếu được thì có thể gợi ý nên nhập chuỗi gì!!
      2. Gửi cho server số đã nhập
      3. Nhận chuỗi ký tự từ server gửi về. Nếu chuỗi đó là “**Chuc mung vi da doan dung”** thì thoát khỏi vòng lặp
   3. Đóng kết nối

# Yêu cầu server phải phản hồi request của client trong vòng không quá 1 giây!!! Nếu client có gợi ý thì mới được điểm tối đa.

**# Trả lời:**

**Dán code server vào bên dưới**

package De1\_TCP;

import java.io.DataInputStream; import java.io.DataOutputStream; import java.io.IOException; import java.net.ServerSocket;

import java.net.Socket;

public class TCPServer { public TCPServer() {

try {

ServerSocket serverSocket = new ServerSocket(5555); System.***out***.println("Server is Startting...");

while (true) {

Socket socket = serverSocket.accept(); System.***out***.println("Connected to " +

socket.getInetAddress().getHostAddress() + "/" + socket.getPort()); Processing serverThread = new Processing(socket); serverThread.start();

}

} catch (IOException e) { e.printStackTrace();

}

}

public static void main(String[] args) { new TCPServer();

}

}

class Processing extends Thread {

private static final String ***STRING*** = "PhamMinhTuanPhamMinhTuanPhamMi"; private Socket socket;

public Processing(Socket socket) { this.socket = socket;

}

*@Override*

public void run() { try (

DataInputStream dis = new DataInputStream(socket.getInputStream()); DataOutputStream dos = new DataOutputStream(socket.getOutputStream())

) {

while (true) {

String clientInput = dis.readUTF(); System.***out***.println("Client input: " + clientInput);

if (!clientInput.matches("^[a-zA-Z]+$")) { dos.writeUTF("Khong dung dinh dang"); continue;

}

int comparisonResult = clientInput.compareTo(***STRING***); if (comparisonResult > 0) {

dos.writeUTF(">");

} else if (comparisonResult < 0) { dos.writeUTF("<");

} else {

dos.writeUTF("Chuc mung vi da doan dung"); break;

}

}

} catch (IOException e) { e.printStackTrace();

} finally {

try {

socket.close();

} catch (IOException e) { e.printStackTrace();

}

}

}

}

# Dán code client vào bên dưới

package De1\_TCP;

import java.io.BufferedReader; import java.io.DataInputStream; import java.io.DataOutputStream; import java.io.InputStreamReader; import java.net.Socket;

public class Client { public Client() {

try {

Socket socket = new Socket("localhost", 5555);

DataOutputStream dos = new DataOutputStream(socket.getOutputStream()); DataInputStream dis = new DataInputStream(socket.getInputStream()); BufferedReader br = new BufferedReader(new

InputStreamReader(System.***in***));

while (true) {

System.***out***.println("Enter a string (ký tự alphabet):"); String inputString = br.readLine();

dos.writeUTF(inputString);

String serverResponse = dis.readUTF(); System.***out***.println("Server response:"); System.***out***.println(serverResponse);

if ("Chuc mung vi da doan dung".equals(serverResponse)) { break;

}

}

socket.close();

} catch (Exception e) { e.printStackTrace();

}

}

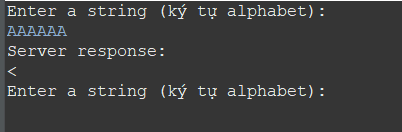
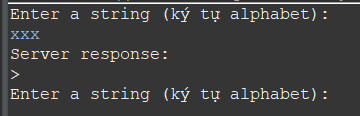
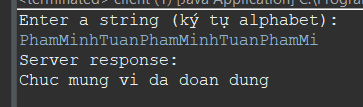
public static void main(String[] args) { new Client();

}

}

}

# Dán các kết quả thực thi vào bên dưới



***Câu 2*** (*2 điểm*): Hãy cho biết nếu chuyển chương trình ở câu 1 qua giao thức UDP thì các công việc cần làm gồm những gì? (Không cần code, chỉ cần mô tả)

**# Trả lời:**

**Ở phía Server**

Khi tạo socket dử dụng DatagramSocket.

Tạo biến receiveData kiểu byte[] để nhận dữ liệu từ gói tin đến , tạo biến DatagramPacket receivePacket = new DatagramPacket(receiveData,receiveData.length) để nhận gói tin từ socket. Nhận gói tin từ phương thức receive() , chuyển dữ liệu kiểu byte của gói tin thành kiểu String. Để gửi gói tin về lại client thì trước hết phải lấy địa chỉ IP và port của bên gửi.

Sau đó chuyển gói tin cần gửi sang kiểu byte[] và sử dụng phương thức send() để gửi gói tin đi.

**Ở phía Client**

Tạo DatagramSocket:

Client cần tạo một DatagramSocket để gửi và nhận gói tin. Nhập Dữ Liệu từ Người Dùng:

Client cần thu thập dữ liệu từ người dùng, chẳng hạn như chuỗi cần gửi đến server. Tạo DatagramPacket và Gửi Gói Tin:

Dữ liệu được đóng gói vào DatagramPacket, kèm theo địa chỉ IP và cổng của server, sau đó được gửi đi qua DatagramSocket.send().

Chờ và Nhận Phản Hồi từ Server:

Client sử dụng DatagramSocket.receive() để chờ và nhận phản hồi từ server.

**Xử Lý Phản Hồi:**

Client xử lý phản hồi nhận được từ server và hiển thị hoặc thực hiện các hành động tương ứng. Kiểm Tra Điều Kiện Dừng:

Nếu đạt được điều kiện dừng (ví dụ: dữ liệu nhập từ người dùng thỏa mãn điều kiện nào đó), client thoát khỏi vòng lặp và đóng kết nối.

***Câu 3*** (5 *điểm*): Trong phần bài tập JSP/Servlet đã nộp

**Đề bài tập**: *Có 1 tính toán lớn (chạy ngầm, ví dụ như xử lý dữ liệu lớn, xử lý video, convert pdf qua doc, xây dựng mô hình học máy, kiểm tra đạo code, đạo văn tự động,...). Khi client gửi thông tin cần thực hiện xử lý, server sẽ đẩy thông tin đó vào 1 hằng đợi để thực hiện. Client sẽ xem kết quả xử lý thông qua account của bản thân* .

Hãy trả lời các câu hỏi sau:

1. Cho biết tên của các thành viên trong nhóm, kể cả bản thân

**# Trả lời:** viết câu trả lời vào bên dưới Nguyễn Hữu Phúc

Hồ Anh Dũng

1. Mô tả chức năng chính mà bản thân đã đóng góp vào trong chương trình.

**# Trả lời:** viết câu trả lời vào bên dưới

viết Bean,DAO,BO chức năng : convert File ,UpLoadfile, DownloadFile.

Viết controller của chức năng :DownloadFileSeverlet,convertFileSeverlet,UpLoadfileSeverlet. viết báo cáo Hướng dẫn cài đặt chương trình

1. *Hãy mô tả cách thức để có thể xử lý một request với thời gian lớn mà không bị “request time out”. Hãy trích tối đa 10 dòng code trong bài tập đã làm thể hiện điều đó! (có thể thêm comment nếu cần, nhưng không được trích quá 10 dòng code)*

**# Trả lời**: viết câu trả lời vào bên dưới

public static void main(String[] args) {

try { CompletableFuture<String> asyncResult = *handleAsyncRequest*(); String result = asyncResult.get(10, *TimeUnit*.***SECONDS***); System.***out***.println("Received result: " + result);

} catch (TimeoutException | InterruptedException | ExecutionException e) { e.printStackTrace();

}}

private static CompletableFuture<String> handleAsyncRequest() { return CompletableFuture.*supplyAsync*(() -> {

try { Thread.*sleep*(5000);

} catch (InterruptedException e) { e.printStackTrace();}

return "Request processed successfully!";

});

Đà Nẵng, ngày 12 tháng 12 năm 2023