A blue logo with a black background

Description automatically generated

BÁO CÁO THỰC HÀNH

**Bài thực hành số 03**

**Môn học:** Lập trình ứng dụng mạng

**Lớp:** NT109.O21.MMCL

**SINH VIÊN THỰC HIỆN:**

|  |  |
| --- | --- |
| **Họ và tên** | **MSSV** |
| Nguyễn Văn Huấn | 21522108 |

# BÁO CÁO CHI TIẾT

## Nội dung 1

### TCP chat

A screen shot of a computer program

Description automatically generated

(Phần client của chat TCP)

Server sẽ là localhost và port sử dụng là port 1234 (nên sử dụng các port trên 1000 vì các port dưới 1000 đã có các dịch vụ khác sử dụng)

Trong khối try:

* + Hiển thị thông báo và ghi log về quá trình kết nối tới máy chủ.
  + Tạo một socket và kết nối tới máy chủ.
  + Ghi log về việc kết nối tới máy chủ.
  + Tạo luồng đầu ra để gửi dữ liệu tới máy chủ và ghi log về việc gửi dữ liệu.
  + Gửi một thông điệp tới máy chủ và ghi log về thông điệp đã gửi.
  + Tạo luồng đầu vào để nhận dữ liệu từ máy chủ và ghi log về việc nhận dữ liệu.
  + Đọc và hiển thị thông điệp được trả về từ máy chủ và ghi log về thông điệp đã nhận.
  + Đóng kết nối tới máy chủ sau khi hoàn thành việc trao đổi thông tin.

Trong khối catch:

* + In ra stack trace của ngoại lệ nếu có bất kỳ lỗi nào xảy ra trong quá trình kết nối.

A screenshot of a computer program

Description automatically generated

(Phần server của chat TCP)

Trong khối try:

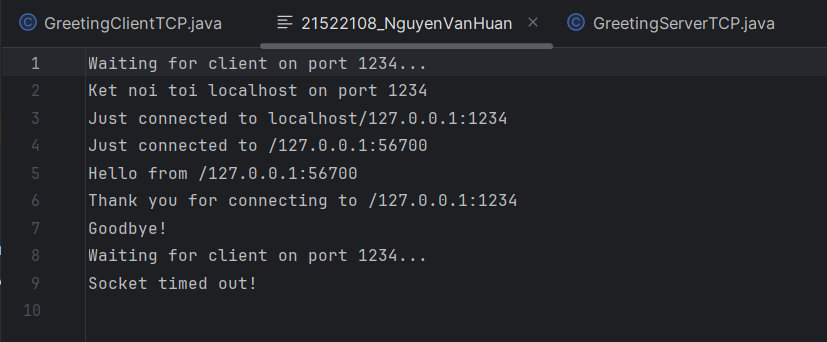
* + Hiển thị thông báo và ghi log về việc đang chờ đợi kết nối từ client trên cổng nào.
  + Chờ đợi và chấp nhận kết nối từ client bằng cách sử dụng phương thức **accept()** của **ServerSocket**. Khi một kết nối được thiết lập, phương thức này trả về một đối tượng **Socket** đại diện cho kết nối với client.
  + Hiển thị thông báo và ghi log về việc kết nối thành công với client.
  + Tạo luồng đầu vào (**DataInputStream**) để nhận dữ liệu từ client.
  + Đọc dữ liệu từ client và hiển thị ra màn hình.
  + Tạo luồng đầu ra (**DataOutputStream**) để gửi dữ liệu đến client.
  + Gửi một thông điệp đến client và ghi log về việc gửi thông điệp này.
  + Đóng kết nối với client bằng cách gọi phương thức **close()** trên đối tượng **Socket**.

Trong khối catch **SocketTimeoutException**:

* + Hiển thị thông báo và ghi log về việc kết nối bị timeout khi chờ đợi client.
  + Kết thúc vòng lặp vô hạn bằng cách sử dụng lệnh **break**.

Trong khối catch **IOException**:

* + In ra stack trace của ngoại lệ để ghi lại lỗi và tiếp tục thực thi chương trình.
  + Kết thúc vòng lặp vô hạn bằng cách sử dụng lệnh **break**.



(file log sau khi thực hiện kết nối TCP, run server trước rồi sau đó đến client)

### b. UDP chat

A computer screen shot of a program code

Description automatically generated

(Phần UDP server)

socket = new DatagramSocket(PORT); System.out.println(("Server started on port "+ PORT)); logger.log("Server started on port "+ PORT); Scanner scanner = new Scanner(System.in);

* + Mở một socket Datagram trên cổng được chỉ định (**PORT**).
  + Hiển thị thông báo và ghi log về việc máy chủ đã được khởi động trên cổng này.
  + Tạo một đối tượng **Scanner** để đọc dữ liệu nhập từ bàn phím.

new Thread(() -> { // Mã xử lý nhận dữ liệu từ client }).start();

* + Bắt đầu một thread mới để xử lý việc nhận dữ liệu từ client. Thread này sẽ chạy một vòng lặp vô hạn để liên tục nhận dữ liệu.

Trong thread xử lý nhận dữ liệu từ client:

* + Nhận dữ liệu từ client bằng cách gọi phương thức **receive()** trên socket.
  + Chuyển đổi dữ liệu nhận được thành chuỗi và hiển thị ra màn hình.
  + Ghi log về dữ liệu nhận được từ client.

new Thread(() -> { // Mã xử lý gửi dữ liệu tới client }).start();

* + Bắt đầu một thread mới để xử lý việc gửi dữ liệu tới client. Thread này cũng chạy một vòng lặp vô hạn để liên tục nhận dữ liệu từ người dùng và gửi nó tới client.

Trong thread xử lý gửi dữ liệu tới client:

* + Đọc dữ liệu từ người dùng nhập vào từ bàn phím.
  + Chuyển đổi dữ liệu thành mảng byte và đóng gói nó vào một DatagramPacket để gửi đi.
  + Gửi DatagramPacket đó đi tới client thông qua socket.

A screenshot of a computer program

Description automatically generated

A computer screen shot of a program code

Description automatically generated  
(Phần UDP client)

static logFile logger = new logFile("21522108\_NguyenVanHuan");

private static int SERVER\_PORT = 5000;

private static int CLIENT\_PORT = 5001;

private static DatagramSocket socket;

* + Khai báo các biến static để định nghĩa cổng của máy chủ và cổng của client, cũng như một đối tượng DatagramSocket để tạo và quản lý kết nối UDP.
  + Khai báo một đối tượng logger để ghi log các sự kiện.

Trong phương thức **main()**:

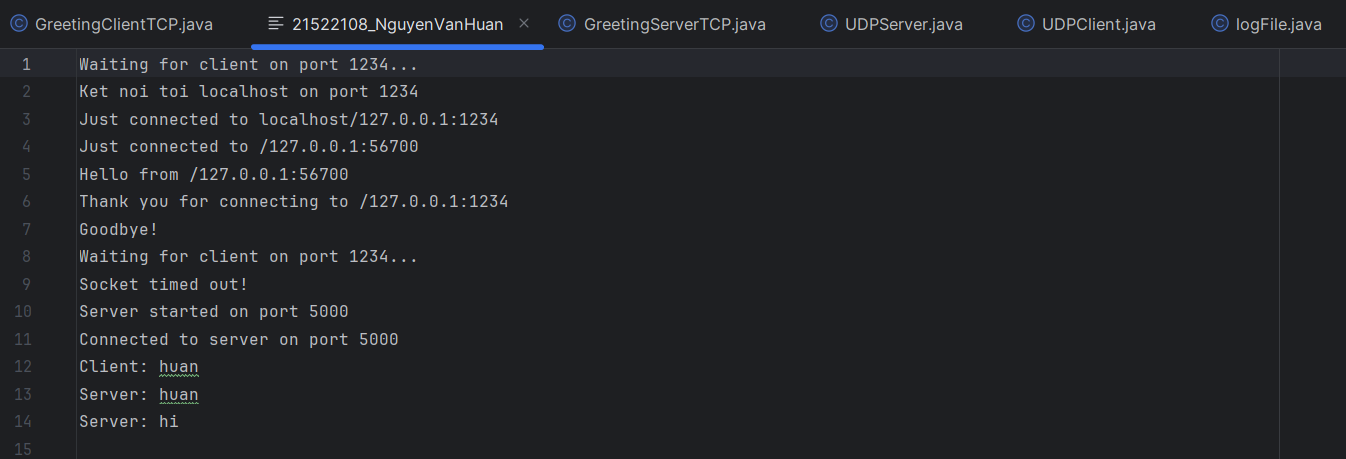
* + Hiển thị thông báo về việc kết nối với server qua cổng đã chỉ định.
  + Khởi tạo một đối tượng DatagramSocket để kết nối với client qua cổng CLIENT\_PORT.
  + Xác định địa chỉ của máy chủ (trong trường hợp này, là "localhost").

Một thread mới được bắt đầu để nhận dữ liệu từ server:

* + Thread này chạy một vòng lặp vô hạn để liên tục nhận dữ liệu từ server.
  + Dữ liệu nhận được được lưu vào một DatagramPacket và sau đó được hiển thị ra màn hình và ghi log.

Một thread mới khác được bắt đầu để gửi dữ liệu tới server:

* + Thread này cũng chạy một vòng lặp vô hạn để liên tục đọc dữ liệu nhập từ người dùng từ bàn phím và gửi nó tới server qua DatagramPacket.



(file log sau khi run UDP Server và UDP Client, thực hiện chat giữa server và client bằng chat UDP)

## Khó khăn

- Phần khởi tạo luồng thread và các thao tác xử lý với thread còn hạn chế, chưa thành thạo

# TÀI LIỆU THAM KHẢO