BÁO CÁO THỰC HÀNH

**Môn học: Lập trình ứng dụng mạng**

**Buổi báo cáo: Lab 03**

**Tên chủ đề: Lập Trình Mạng Với Java Socket**

*GVHD: Trần Mạnh Hùng*

*Ngày thực hiện: 14/04/2024*

**THÔNG TIN CHUNG:**

*(Liệt kê tất cả các thành viên trong nhóm)*

Lớp: NT109.O21.MMCL

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Họ và tên** | **MSSV** | **Email** |
| 1 | Nguyễn Văn Anh Tuấn | 21522757 | 21522757@gm.uit.edu.vn |

1. **ĐÁNH GIÁ KHÁC:**

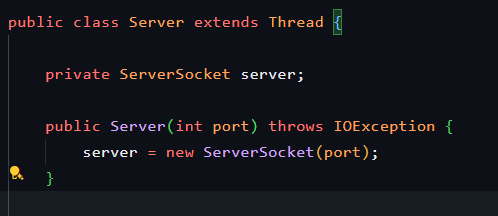
|  |  |
| --- | --- |
| **Nội dung** | **Kết quả** |
| Tổng thời gian thực hiện bài thực hành trung bình | 5 tiếng |
| Link Video thực hiện  *(nếu có)* |  |
| Ý kiến *(nếu có)*  + Khó khăn  + Đề xuất … | Không có |
| Điểm tự đánh giá | 10/10 |

BÁO CÁO CHI TIẾT

**1) TCP Socket:**

a) Server:

Chương trình **Server** sử dụng lớp Socket để lắng nghe các client ở trên một port nhất định.



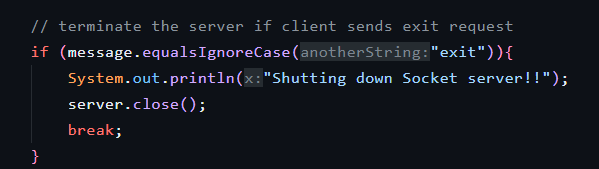
Hình 1.1

Hàm **run()** xử lý việc lắng nghe và ServerSocket để accept các client. Sử dụng **ObjectInputStream** để xử lý các message được gửi đến và **ObjectOutputStream** để gửi message đến các client.



Hình 1.2

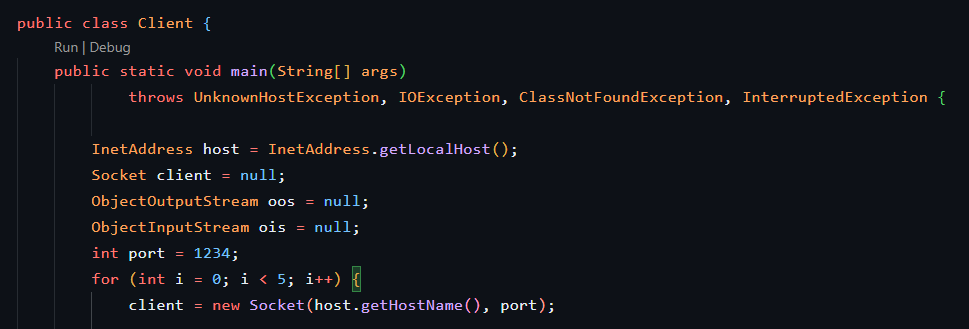
Sử dụng vòng lặp while(true) chỉ dừng khi client gửi tin nhắn có giá trị là “exit”.



Hình 1.3

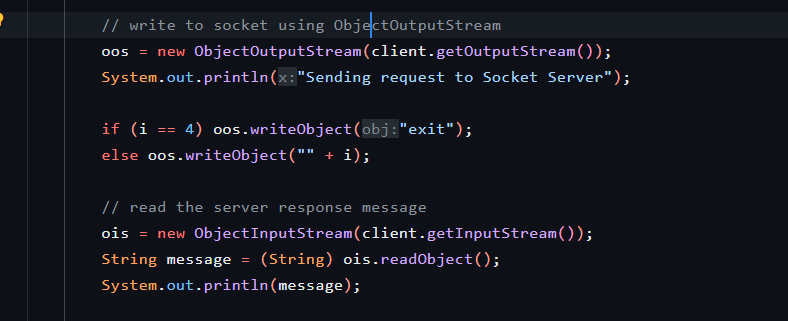
b) Client:

Chương trình **Client** sử dụng lớp Socket để kết nối tới host socket server với số port được chỉ định.



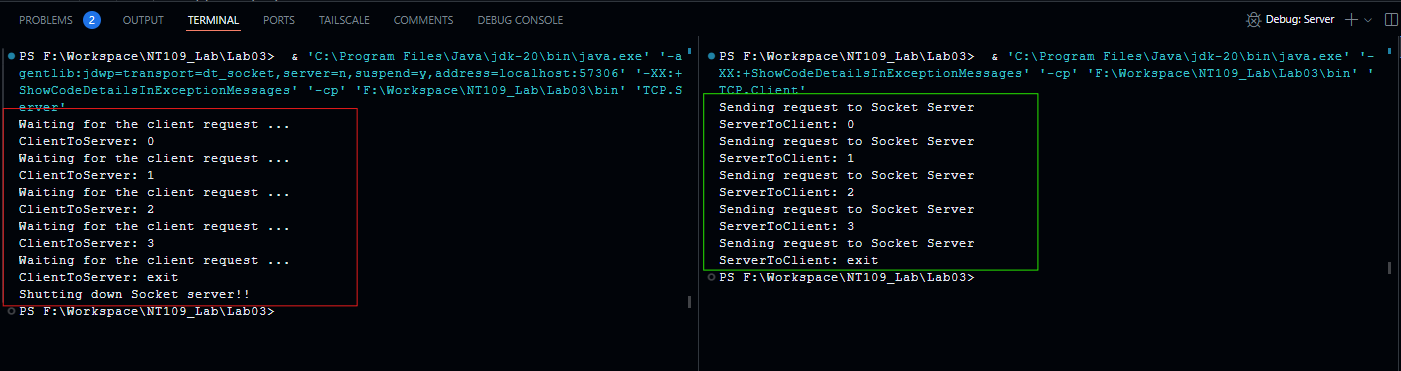
Hình 1.4

Sử dụng **ObjectInputStream** để xử lý các message được gửi đến, và **ObjectOutputStream** để xử lý các message được gửi đi. Ở ví dụ dưới đây khi tới vòng lặp cuối cùng sẽ gửi tin nhắn mang giá trị “exit” để đóng **Socket Server** và ngắt kết nối. Lý do phải khởi tạo **oos** và **ois** sau mỗi vòng lặp tại vì sau khi **accept()** Server hiện tại chỉ nhận Input, read() từ phía client. Phải khởi tạo **oos** trước sau đó mới đến **ois** nếu không sẽ gây ra lỗi cả 2 phía đều lắng nghe.



Hình 1.5

* DEMO:

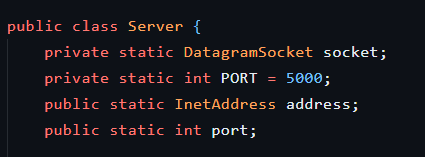


Hình 1.6

**2) UDP Socket:**

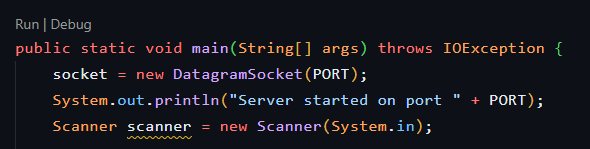
a) Server:

Sử dụng **DatagramSocket** để gửi và nhận các gói tin UDP. Biến **PORT** là số port của server. Biến **address** và **port** được sử dụng để quản lý địa chỉ và port của client kết nối đến.



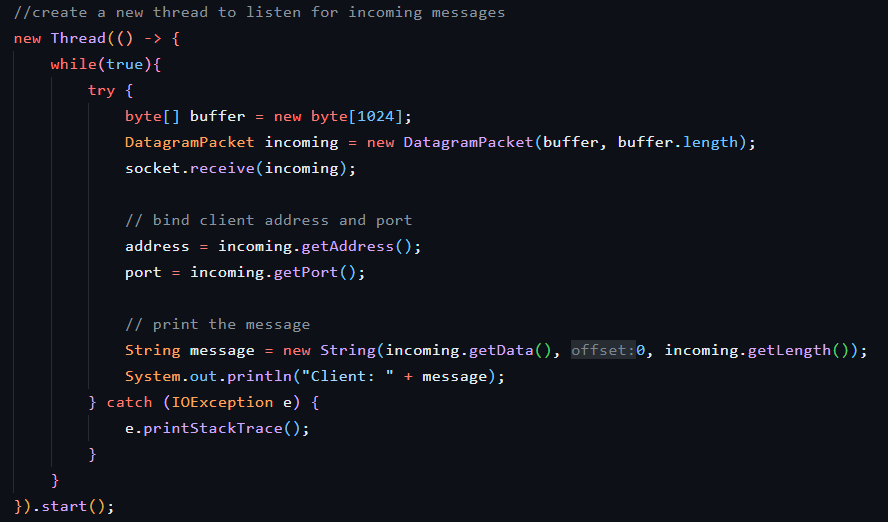
Hình 2.1

Khởi tạo socket với số port chỉ định để lắng nghe các kết nối.



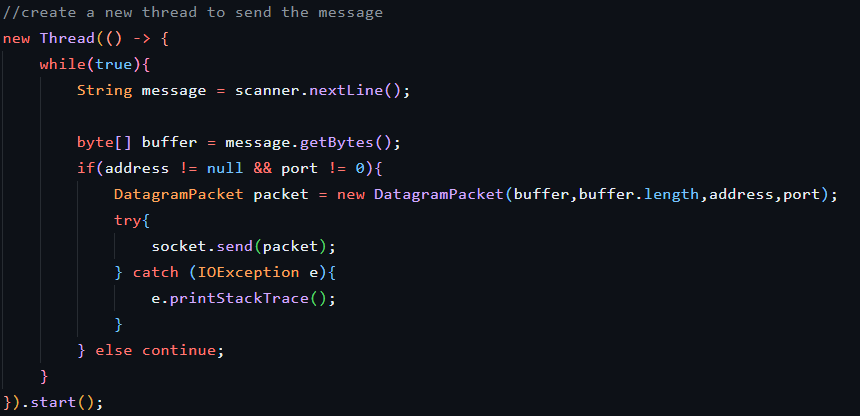
Hình 2.2

Tạo một thread mới để lắng nghe các gói tin UDP được gửi đến. Sau khi nhận được gói tin từ phía client gán địa chỉ và port của client cho 2 biến **address** và **port**. Cuối cùng in tin nhắn nhận được ra console.



Hình 2.3

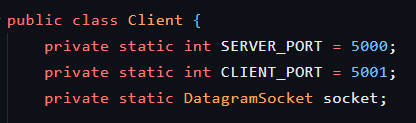
Tạo một thread mới để gửi các gói tin UDP đến client thông qua địa chỉ và số port của client gửi đến. Nếu việc gửi tin từ Server xảy ra trước khi có client kết nối đến sẽ xảy ra lỗi vì **address** và **port** chỉ được gán sau khi client gửi tin nhắn. Để tránh trường hợp này em thêm một lệnh điều kiện cho phép server gửi tin nhắn khi **address và port khác null**.



Hình 2.4

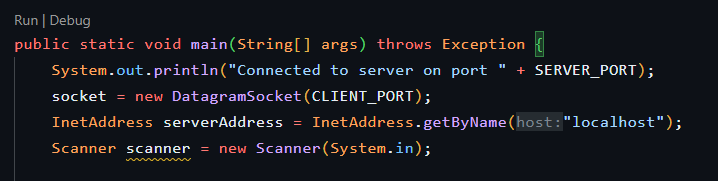
b) Client:

Sử dụng **DatagramSocket** để gửi và nhận các gói tin UDP. Biến **SERVER\_PORT** là số cổng của server và **CLIENT\_PORT** là số cổng của client.



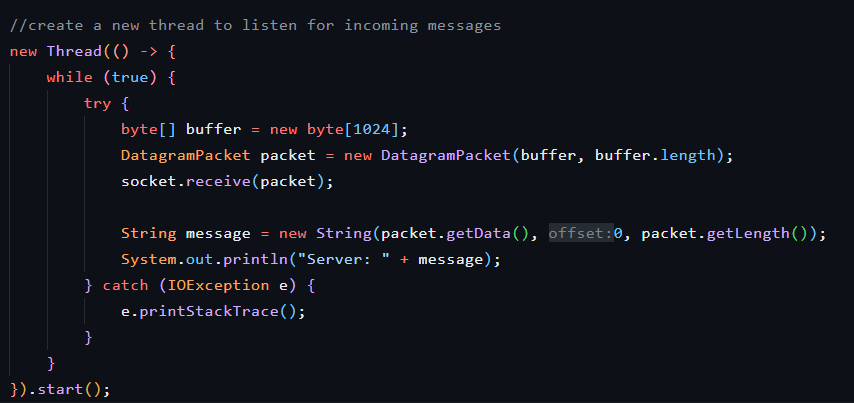
Hình 2.5

Thiết lập kết nối tới server bằng giao thức UDP.

****

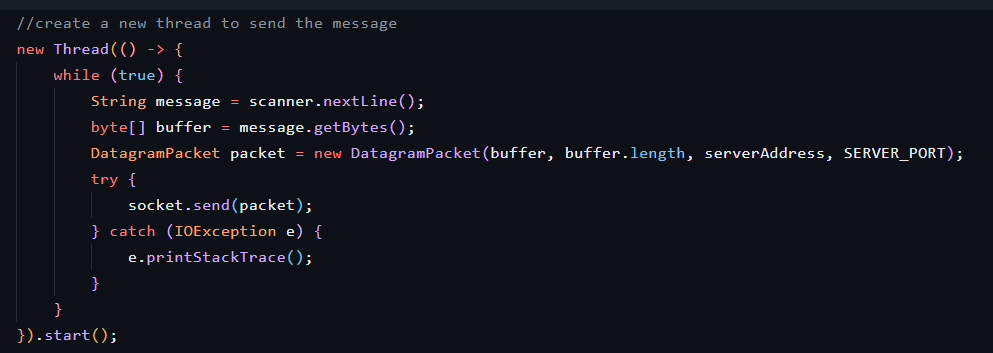
Hình 2.6

Tạo một thread mới dùng cho việc lắng nghe các gói tin UDP gửi tới từ Server. Sau đó in tin nhắn ra console.



Hình 2.7

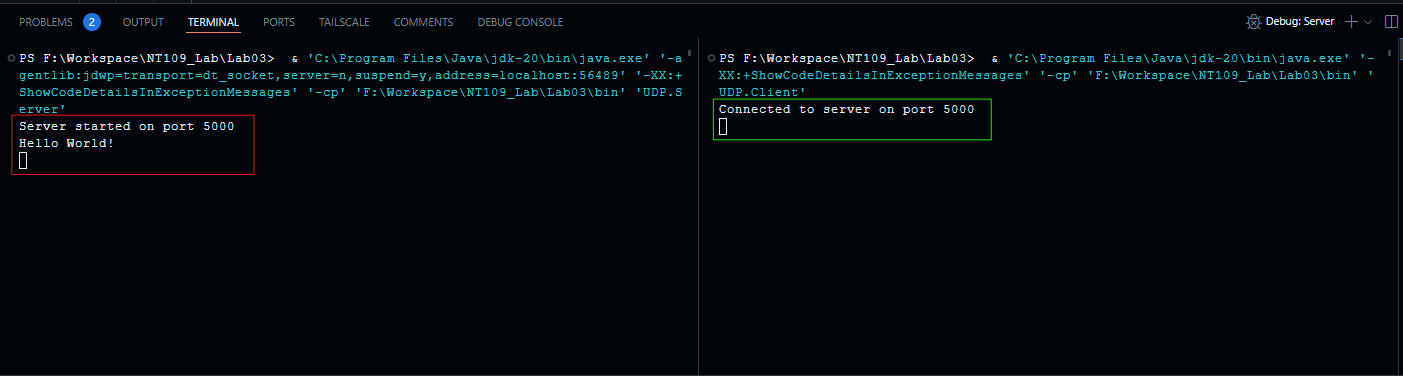
Tạo một thread mới dùng cho việc gửi các gói tin UDP đến Server từ địa chỉ và số port có sẵn.



Hình 2.8

* DEMO:

Trường hợp server gửi tin nhắn trước:



Hình 2.9

Trường hợp Client gửi tin nhắn trước:



Hình 2.10

**HẾT**