

**ĐẠI HỌC QUỐC GIA HÀ NỘI**  
**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN**  
**KHOA TOÁN - CƠ - TIN HỌC**  
**---0000000---**



**BÁO CÁO**  
**MÔN CÔNG NGHỆ PHẦN MỀM**

**Chủ đề: Mô hình Server-Client đơn giản**

**Giảng viên : PGS.TS Bùi Việt Hà**

**Thành viên trong nhóm : Lê Tuyết Anh - 20001883**

**Đỗ Thị Bích Thủy - 20001979**

**Vũ Thị Ngọc Quyên - 20001969**

**Đinh Phương Linh - 20001941**

**Hà Nội, 09 - 2023**

## MỤC LỤC

<b>A. Giới thiệu.....</b>	<b>1</b>
Mục đích.....	1
Một số khái niệm.....	1
Phạm vi.....	2
Bố cục tài liệu.....	2
<b>B. Yêu cầu.....</b>	<b>2</b>
Yêu cầu về môi trường chạy.....	2
Yêu cầu về giao diện.....	2
<b>C. Sơ đồ chức năng (Use-case).....</b>	<b>3</b>
<b>D. Danh sách chức năng:.....</b>	<b>4</b>
<b>E. Yêu cầu phi chức năng.....</b>	<b>5</b>
<b>F. Test-case.....</b>	<b>6</b>
Trường hợp thông.....	6
Trường hợp lỗi.....	7
<b>G. Tài liệu tham khảo.....</b>	<b>7</b>

## A. Giới thiệu

### Mục đích

Phát triển một server để tạo nên một ứng dụng trò chuyện với mô hình server - client (mô hình khách - chủ). Ứng dụng cho người dùng (client) tạo tài khoản và dùng nó để tham gia vào các cuộc trò chuyện, gửi message. Dự án cũng là một ví dụ về một web-server cơ bản giúp hiểu rõ hơn về cơ chế hoạt động cũng như cách thức xây dựng, sử dụng web-server cùng đó là tìm hiểu về lập trình bằng socket.

### Một số khái niệm

**Mô hình client-server** (mô hình khách - chủ) là cấu trúc ứng dụng phân tán, ở đó công việc được phân chia giữa bên cung cấp tài nguyên hoặc dịch vụ (server) và bên yêu cầu dịch vụ (client) hoặc có thể nói Client là nguồn đưa ra các yêu cầu và bên Server phải phục vụ theo nó.

Trong đó hai phần chính là server và client :

- Client là những máy khách, máy trạm và cũng là nơi thực hiện việc gửi các request đến với server. Tại đây client giao tiếp với người dùng, server và cả môi trường bên ngoài. Khi nhận được kết quả từ server, chúng sẽ tổ chức và thể hiện các kết quả.
- Server sẽ thực hiện xử lý các yêu cầu được gửi đến từ client. Sau khi xử lý và hoàn thiện nó sẽ trả kết quả thu được về cho client. Sau đó client tiếp tục xử lý các kết quả này và gửi đến người dùng. Server thực hiện giao tiếp, tiếp nhận thông tin yêu cầu dưới dạng query string (các xâu ký tự). Sau khi phân tích sẽ thực hiện xử lý dữ liệu và gửi về kết quả.

Vai trò của server:

- Server như là một nhà cung cấp dịch vụ (cơ sở dữ liệu, in ấn, truyền file, hệ thống,...) cho các clients yêu cầu tới khi cần
- Các ứng dụng server cung cấp các dịch vụ mang tính chức năng để hỗ trợ cho hoạt động trên các máy clients đạt hiệu quả hơn.
- Để đảm bảo tính an toàn trên mạng cho nên server này còn có vai trò như là một nhà quản lý toàn bộ quyền truy cập dữ liệu của các máy clients, nói cách khác đó là vai trò quản trị mạng.

**Socket** là giao diện lập trình ứng dụng mạng được dùng để truyền và nhận dữ liệu trên internet. Giữa hai chương trình chạy trên mạng cần có một liên kết giao tiếp hai chiều, hay còn gọi là two-way communication để kết nối 2 process trò chuyện với nhau. Điểm cuối (endpoint) của liên kết này được gọi là socket.

### **Phạm vi**

Trong khuôn khổ thời gian, đề tài và năng lực, client-server mang tính chất demo, chỉ sử dụng trên một máy local, không public lên mạng ngoài.

### **Bộ cục tài liệu**

Tài liệu gồm 6 phần chính:

Phần 1. Giới thiệu tổng quan về sản phẩm và tài liệu

Phần 2. Các yêu cầu cụ thể về môi trường chạy và giao diện.

Phần 3. Sơ đồ chức năng : minh họa chức năng server bằng biểu đồ

Phần 4. Danh sách chức năng: liệt kê các chức năng của server và mô tả

Phần 5. Yêu cầu phi chức năng: hướng đi có thể tiếp tục phát triển

Phần 6. Test-case: Chạy demo các kịch bản kiểm thử.

## **B. Yêu cầu**

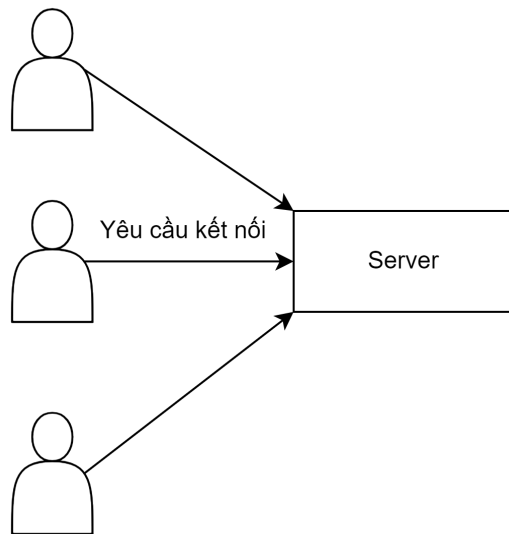
### **Yêu cầu về môi trường chạy**

Chạy demo trên terminal, chương trình viết bằng java nên cần cài JDK để biên dịch, thực thi, chạy thử.

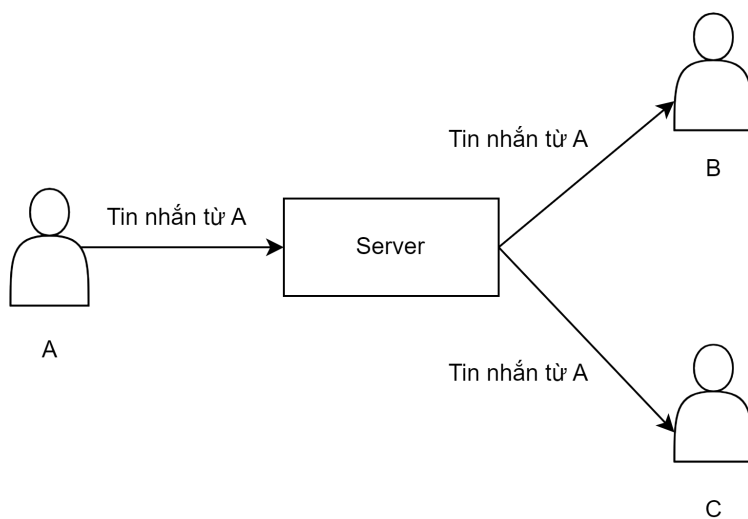
### **Yêu cầu về giao diện**

Không yêu cầu UI/UX, chương trình client-server mang tính chất demo để hiểu luồng chạy và bản chất làm thế nào một server cung cấp service cho các client.

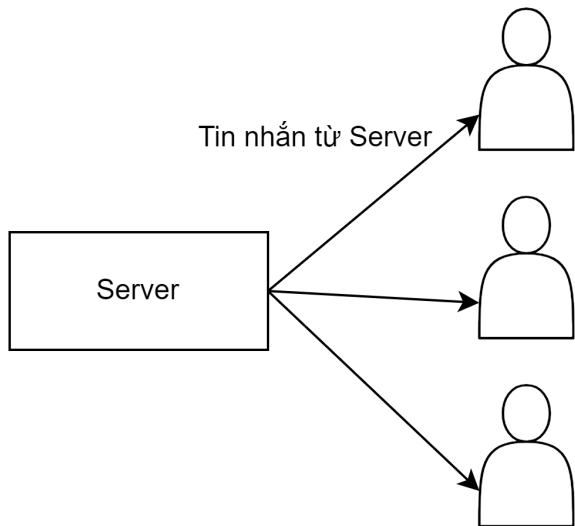
### C. Sơ đồ chức năng (Use-case)



Server luôn chạy và lắng nghe từ port 15797. Các client kết nối đến server bằng cách nhập tên và gửi kết nối đến server. Server nhận yêu cầu và thêm vào danh sách các socket



Người dùng A gửi tin nhắn đến Server, Server chuyển tiếp đến những người dùng còn lại trong danh sách



Server gửi tin nhắn đến toàn bộ người dùng trong danh sách

#### D. Danh sách chức năng:

##### Server

Khởi tạo Server Socket và lắng nghe kết nối: Trong phương thức **execute()**, server sử dụng *ServerSocket* để khởi tạo một máy chủ và lắng nghe các yêu cầu kết nối từ các máy khách trên một cổng cụ thể (port). Một luồng *WriteServer* cũng được khởi tạo để cho phép server gửi tin nhắn đến tất cả các máy khách.

Xử lý kết nối từ các máy khách: Khi một máy khách kết nối đến server thông qua **server.accept()**, một socket mới được tạo để xử lý kết nối với máy khách này. Thông tin về socket này được lưu trữ trong **ArrayList<Socket> listSK** để theo dõi tất cả các kết nối đến server.

Đọc dữ liệu từ máy khách: Mỗi kết nối máy khách được xử lý bởi một luồng *ReadServer*. Luồng này đọc dữ liệu từ máy khách thông qua *DataInputStream* và sau đó chuyển dữ liệu này cho tất cả các máy khách khác thông qua luồng *WriteServer*. Nếu tin nhắn chứa từ khóa "exit", máy khách sẽ bị ngắt kết nối và socket của nó sẽ được loại bỏ khỏi danh sách.

Gửi tin nhắn từ server đến các máy khách: Luồng *WriteServer* cho phép server gửi tin nhắn từ người dùng (người quản trị server) đến tất cả các máy khách hiện đang kết nối thông qua *DataOutputStream*. Tin nhắn này sẽ có định dạng "Server: [tin nhắn từ người quản trị]".

Thông báo khi kết nối bị đóng: Khi một kết nối máy khách bị đóng hoặc có lỗi xảy ra, server sẽ đóng socket tương ứng và loại bỏ nó khỏi danh sách.

## **Client**

Kết nối đến máy chủ: *Client* tạo một đối tượng *Socket* để kết nối đến máy chủ bằng địa chỉ IP và cổng được chỉ định trong *InetAddress* host và int port.

Xác thực và gửi tên người dùng: Trước khi tham gia trò chuyện, client yêu cầu người dùng nhập tên của họ thông qua Scanner. Tên người dùng sau đó được gửi đến máy chủ thông qua *WriteClient* để xác định danh tính của client.

Đọc dữ liệu từ máy chủ: Client sử dụng luồng *ReadClient* để lắng nghe và đọc dữ liệu từ máy chủ thông qua *DataInputStream*. Bất cứ khi nào có tin nhắn mới từ máy chủ, nó sẽ hiển thị tin nhắn này trên giao diện của client.

Gửi tin nhắn đến máy chủ: Client sử dụng luồng *WriteClient* để cho phép người dùng nhập tin nhắn từ bàn phím thông qua Scanner, sau đó gửi tin nhắn này đến máy chủ thông qua *DataOutputStream*. Tin nhắn này bao gồm tên của người dùng và nội dung tin nhắn.

Xử lý mất kết nối: Client cũng đã cài đặt xử lý cho trường hợp mất kết nối với máy chủ. Nếu kết nối bị mất, client sẽ hiển thị thông báo và thoát ứng dụng.

## **E. Yêu cầu phi chức năng**

Các yêu cầu phi chức năng của hệ thống có thể phát triển trong tương lai:

### **Bảo mật:**

Xác thực người dùng: Yêu cầu người dùng đăng nhập trước khi tham gia vào cuộc trò chuyện. Điều này giúp xác minh danh tính của người dùng và đảm bảo rằng chỉ có người dùng được cho phép mới có thể tham gia.

Chia nhóm người dùng: Giới hạn những người được tham gia cuộc trò chuyện.

Kiểm tra dữ liệu đầu vào: Luôn kiểm tra dữ liệu đầu vào từ người dùng để ngăn chặn các cuộc tấn công như CSRF.

### **Tối ưu và cải thiện cho hệ thống**

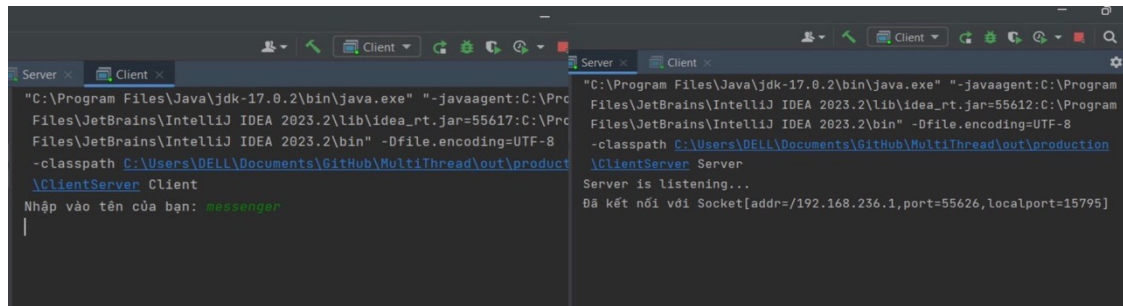
Làm giao diện bắt mắt cho người dùng

Lưu tin nhắn đã được gửi

## F. Test-case

### Trường hợp thông

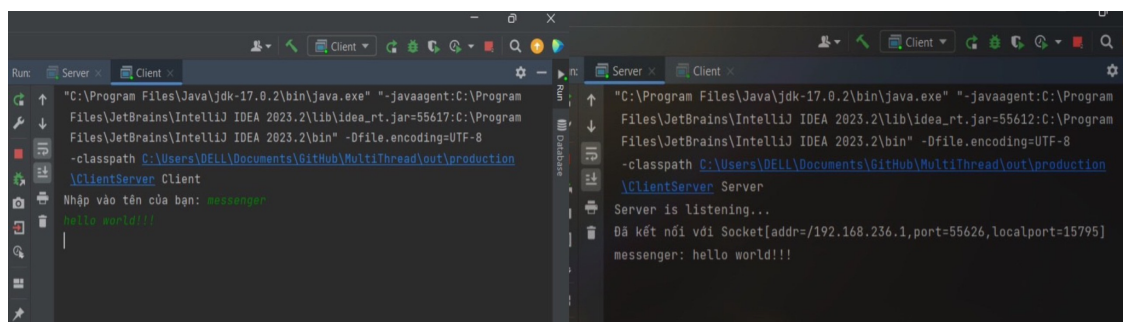
Kiểm tra kết nối: Đảm bảo rằng client có thể kết nối thành công với server.



```
Server: "C:\Program Files\Java\jdk-17.0.2\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2023.2\lib\idea_rt.jar=55617:C:\Program Files\JetBrains\IntelliJ IDEA 2023.2\bin" -Dfile.encoding=UTF-8 -classpath C:\Users\DELL\Documents\GitHub\MultiThread\out\production\ClientServer Server
Server is listening...
Đã kết nối với Socket[addr=/192.168.236.1,port=55626,localport=15795]

Client: "C:\Program Files\Java\jdk-17.0.2\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2023.2\lib\idea_rt.jar=55612:C:\Program Files\JetBrains\IntelliJ IDEA 2023.2\bin" -Dfile.encoding=UTF-8 -classpath C:\Users\DELL\Documents\GitHub\MultiThread\out\production\ClientServer Client
Nhập vào tên của bạn: messenger
```

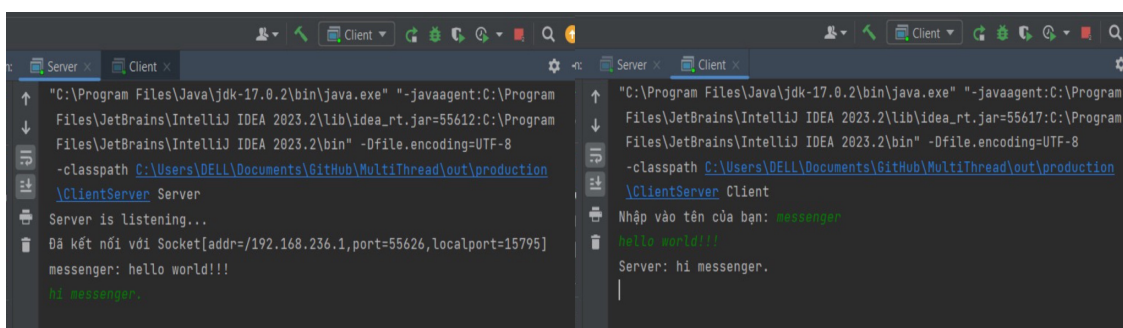
Kiểm tra gửi tin nhắn từ client đến server: Đảm bảo rằng client có thể gửi tin nhắn đến server và server có thể nhận tin nhắn này.



```
Server: "C:\Program Files\Java\jdk-17.0.2\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2023.2\lib\idea_rt.jar=55617:C:\Program Files\JetBrains\IntelliJ IDEA 2023.2\bin" -Dfile.encoding=UTF-8 -classpath C:\Users\DELL\Documents\GitHub\MultiThread\out\production\ClientServer Server
Server is listening...
Đã kết nối với Socket[addr=/192.168.236.1,port=55626,localport=15795]
messenger: hello world!!!

Client: "C:\Program Files\Java\jdk-17.0.2\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2023.2\lib\idea_rt.jar=55612:C:\Program Files\JetBrains\IntelliJ IDEA 2023.2\bin" -Dfile.encoding=UTF-8 -classpath C:\Users\DELL\Documents\GitHub\MultiThread\out\production\ClientServer Client
Nhập vào tên của bạn: messenger
hello world!!!
```

Kiểm tra phát tin nhắn từ server cho tất cả các client: Đảm bảo rằng server có thể gửi tin nhắn đến tất cả các client đã kết nối.



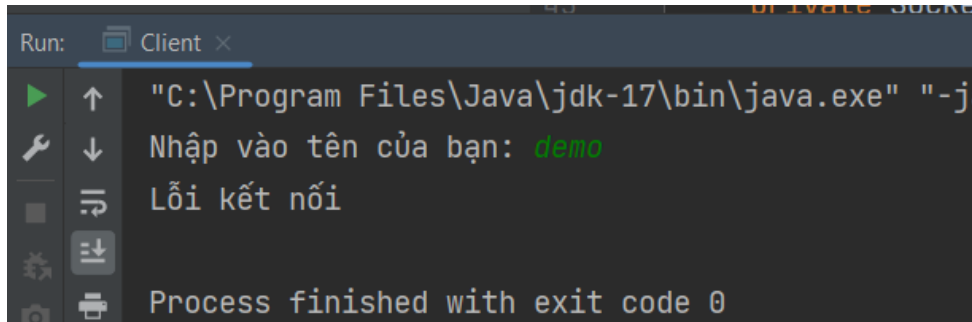
```
Server: "C:\Program Files\Java\jdk-17.0.2\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2023.2\lib\idea_rt.jar=55617:C:\Program Files\JetBrains\IntelliJ IDEA 2023.2\bin" -Dfile.encoding=UTF-8 -classpath C:\Users\DELL\Documents\GitHub\MultiThread\out\production\ClientServer Server
Server is listening...
Đã kết nối với Socket[addr=/192.168.236.1,port=55626,localport=15795]
messenger: hello world!!!
Server: hi messenger.

Client: "C:\Program Files\Java\jdk-17.0.2\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2023.2\lib\idea_rt.jar=55612:C:\Program Files\JetBrains\IntelliJ IDEA 2023.2\bin" -Dfile.encoding=UTF-8 -classpath C:\Users\DELL\Documents\GitHub\MultiThread\out\production\ClientServer Client
Nhập vào tên của bạn: messenger
hello world!!!
Server: hi messenger.
```



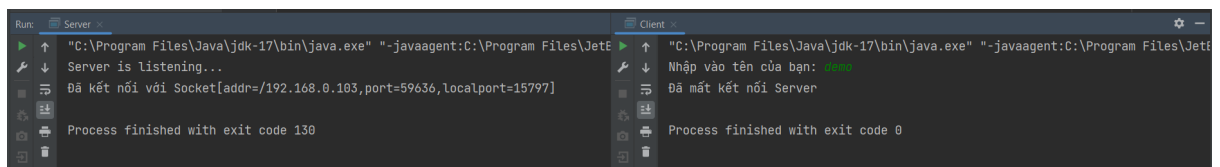
## Trường hợp lỗi

Kiểm tra kết nối thất bại: Thử kết nối từ client đến server sử dụng cổng không tồn tại hoặc server không hoạt động. Đảm bảo rằng client xử lý lỗi kết nối không thành công.



```
Run: Client x
"C:\Program Files\Java\jdk-17\bin\java.exe" "-ja
Nhập vào tên của bạn: demo
Lỗi kết nối
Process finished with exit code 0
```

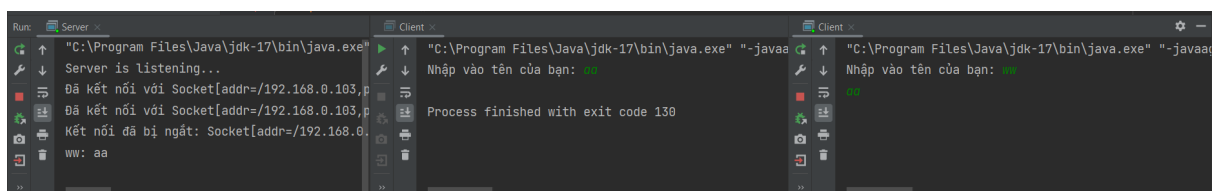
Trong quá trình sử dụng Client mất kết nối với Server, Client thông báo cho người dùng và dừng chương trình.



```
Server:
"C:\Program Files\Java\jdk-17\bin\java.exe" "-javaagent:C:\Program Files\JetE
Server is listening...
Đã kết nối với Socket[addr=/192.168.0.103,port=59636,localport=15797]
Process finished with exit code 130

Client:
"C:\Program Files\Java\jdk-17\bin\java.exe" "-javaagent:C:\Program Files\JetE
Nhập vào tên của bạn: demo
Đã mất kết nối Server
Process finished with exit code 0
```

Client ngắt kết nối đến server, server loại bỏ kết nối:



```
Server:
"C:\Program Files\Java\jdk-17\bin\java.exe"
Server is listening...
Đã kết nối với Socket[addr=/192.168.0.103,p
Kết nối đã bị ngắt: Socket[addr=/192.168.0.
ww: aa

Client:
"C:\Program Files\Java\jdk-17\bin\java.exe" "-javaa
Nhập vào tên của bạn: demo
Process finished with exit code 130

Client:
"C:\Program Files\Java\jdk-17\bin\java.exe" "-javaa
Nhập vào tên của bạn: demo
```

## G. Tài liệu tham khảo

Robert C. Martin, *Clean code: a handbook of agile software craftsmanship*, Phần Appendix trang 317-346. Publisher. Pearson; 1st edition (August 1, 2008)

<https://vi.wikipedia.org/wiki>

<https://viettuts.vn/lap-trinh-mang-voi-java>