TRƯỜNG ĐẠI HỌC KINH BẮC
KHOA CÔNG NGHỆ THÔNG TIN

ĐỀ TÀI

MÔ PHỎNG CỜ VUA

Môn: Lập trình mạng

Giáo viên hướng dẫn: Nguyễn Thị Mười Phương

Sinh viên:

1. Vũ Thị Ngọc Mai (08D4800035)
2. Nguyễn Bá Tới (08D4800061)

Bắc Ninh, 2021

Mục lục:

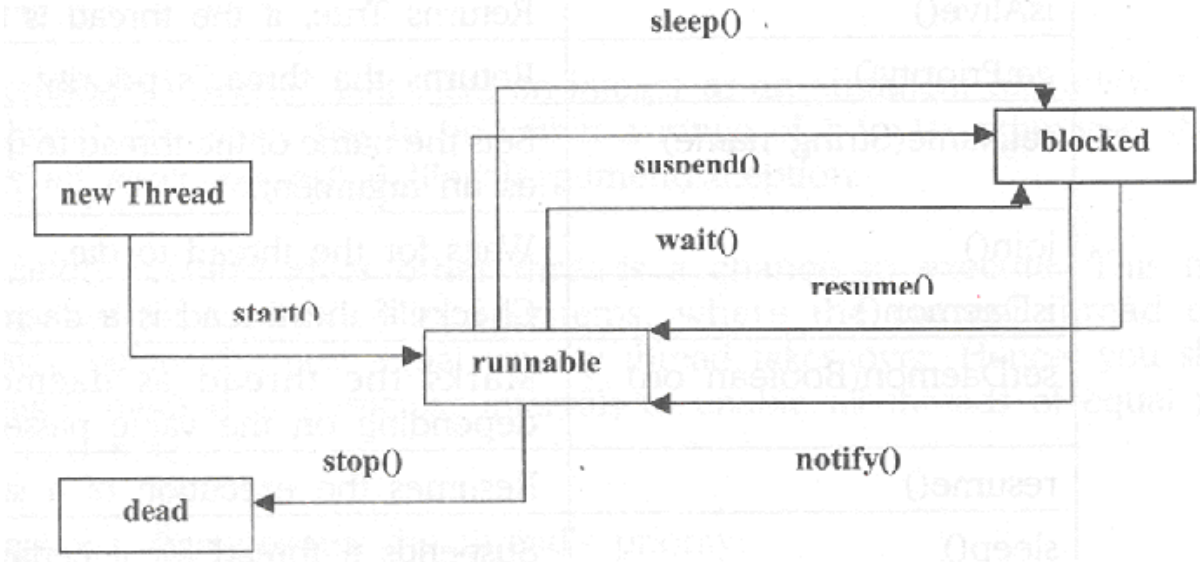
PHẦN I: NGUYÊN LÝ HỆ ĐIỀU HÀNH

TIÊU ĐỀ: TÌM HIỂU ĐỒNG BỘ HOÁ XỬ LÝ ĐỒNG THỜI (CONCURRENT) VÀ GIẢI QUYẾT BÀI TOÁN READERS/WRITERS.

CHƯƠNG 1. CƠ SỞ LÝ THUYẾT

1. Thread và đồng bộ hoá xử lý đồng thời.

- Process (tiến trình): Tiến trình là một chương trình đang xử lý, sỡ hữu một con trỏ lệnh, tập các thanh ghi và các biến. Để hoàn thành tác vụ của mình, một tiến trình có thể cần đến một số tài nguyên – như CPU, bộ nhớ chính, các tập tin và thiết bị nhập/xuất.
- Thread (luồng) là một đơn vị xử lý độc lập của máy tính có thể thực hiện một công việc riêng biệt và có thể xem là một tiến trình con. Một process có thể chứa nhiều thread khác nhau (multithreading). Các thread trong cùng một process chia sẻ chung các tài nguyên như vùng nhớ.
- Vòng đời của thread:



Hình 1: Vòng đời của thread

- Xử lý đồng thời (concurrent): Là quá trình các thread chạy độc lập với nhau trong các khoản thời gian chồng chéo để xử lý các yêu cầu riêng biệt. Các yêu cầu đó có thể được dùng để hoàn thành một yêu lớn hơn.
- Đồng bộ hoá xử lý đồng thời: Khi chạy đồng thời, các thread có thể sử dụng chung các tài nguyên. Để tránh tình trạng xung đột giữa các luồng xử lý ta

phải đồng bộ hoá các thread, tức là chỉ cho phép lần lượt các thread truy cập và sử dụng tài nguyên trong một thời điểm nhất định.

- Semaphore: Là một kernel resource cung cấp một sychronization service. Là một cấu trúc điều khiển đa luồng dựa trên việc block các luồng theo một điều kiện nhất định, nó được xem như là một chiếc chìa khoá để yêu cầu quyền sử dụng một tài nguyên nào đó. Một semaphore sẽ có hai trạng thái là bận và rảnh. Khi một thread yêu cầu lấy chìa khoá để sử dụng tài nguyên, nếu chìa khoá đó đang ở trạng thái rảnh thì thread đó sẽ được cấp quyền sử dụng tài nguyên. Ngược lại, nếu đang ở trạng thái bận, tức là chìa khoá đang được sử dụng bởi một thread khác thì thread vừa yêu cầu sẽ bị block cho đến khi nào chìa khoá trở về trạng thái rảnh.

2. Bài toán Readers/Writers.

2.1 Giới thiệu bài toán:

- Bài toán readers-writers: bài toán yêu cầu xử lý vấn đề khi nhiều thread sử dụng chung một tài nguyên (có thể là đọc hoặc ghi) với ràng buộc: Khi một tiến trình đang thực hiện việc ghi ở tài nguyên thì không một tiến trình nào khác có thể sử dụng tài nguyên đó. Nói cách khác, tài nguyên đó có thể được sử dụng bởi hai hay nhiều reader khác nhau nhưng chỉ được sử dụng bởi một writer.
- Bài toán readers-writers có 3 biến thể:
  - Thứ nhất: Khi hai hoặc nhiều reader yêu cầu quyền sử dụng một tài nguyên, chỉ một reader có quyền sử dụng. Tuy nhiên, do các reader không sửa đổi dữ liệu nên cần giải quyết vấn đề không để bất kỳ một reader nào phải chờ để sử dụng tài nguyên khi tài nguyên đó đang được sử dụng bởi một reader.
  - Thứ hai: Khi một reader R1 đang có quyền truy cập, một writer W đang chờ R1 thực hiện xong để nhận quyền truy cập. Nếu một reader R2 sau đó cũng yêu cầu quyền truy cập và áp dụng biến thể thứ nhất, W sẽ phải tiếp tục đợi R1 và R2. Vì vậy phải giải quyết vấn đề không một writer nào phải đợi quyền access tài nguyên một khi đã được thêm vào hàng đợi.
  - Thứ ba: Hai biến thể trước gây ra tình trạng hoặc là reader phải đợi hoặc là writer phải đợi trong thời gian rất dài. Vì vậy biến thể thứ ba yêu cầu phải giải quyết vấn đề Các reader, writer không được đợi lâu hơn một lượng thời gian xác định. Các readers/writers nào đến trước sẽ được sử dụng tài nguyên trước.

2.2 Giải quyết bài toán:

- Bài toán được giải quyết theo nguyên tắc ai đến trước sẽ được quyền sử dụng tài nguyên trước. Đối với các Reader, nếu có một reader đang sử dụng tài nguyên mà tiếp theo sau là các Reader khác đang chờ thì cho phép các Reader đó truy cập tài nguyên (Do các Reader không làm thay đổi tài nguyên trong quá trình sử dụng). Như vậy sẽ giải quyết được vấn đề các reader đến sau nhưng được sử dụng tài nguyên trước các writer đến trước và vấn đề các Reader chờ đợi một Reader khác không cần thiết.

CHƯƠNG 2. PHÂN TÍCH THIẾT KẾ HỆ THỐNG

1. Yêu cầu bài toán

- Yêu cầu đặt ra của bài toán là cấp phát quyền truy cập và sử dụng tài nguyên cho các thread một cách hợp lý, sao cho không xảy ra các tình trạng sau:
  - Chỉ có một writer sử dụng tài nguyên tại một thời điểm nhất định.
  - Các reader có thể sử dụng chung tài nguyên tại một thời điểm nhất định.
  - Writer/Reader nào yêu cầu sử dụng tài nguyên trước sẽ được sử dụng trước.
  - Các Reader không đợi lẫn nhau khi yêu cầu sử dụng tài nguyên.

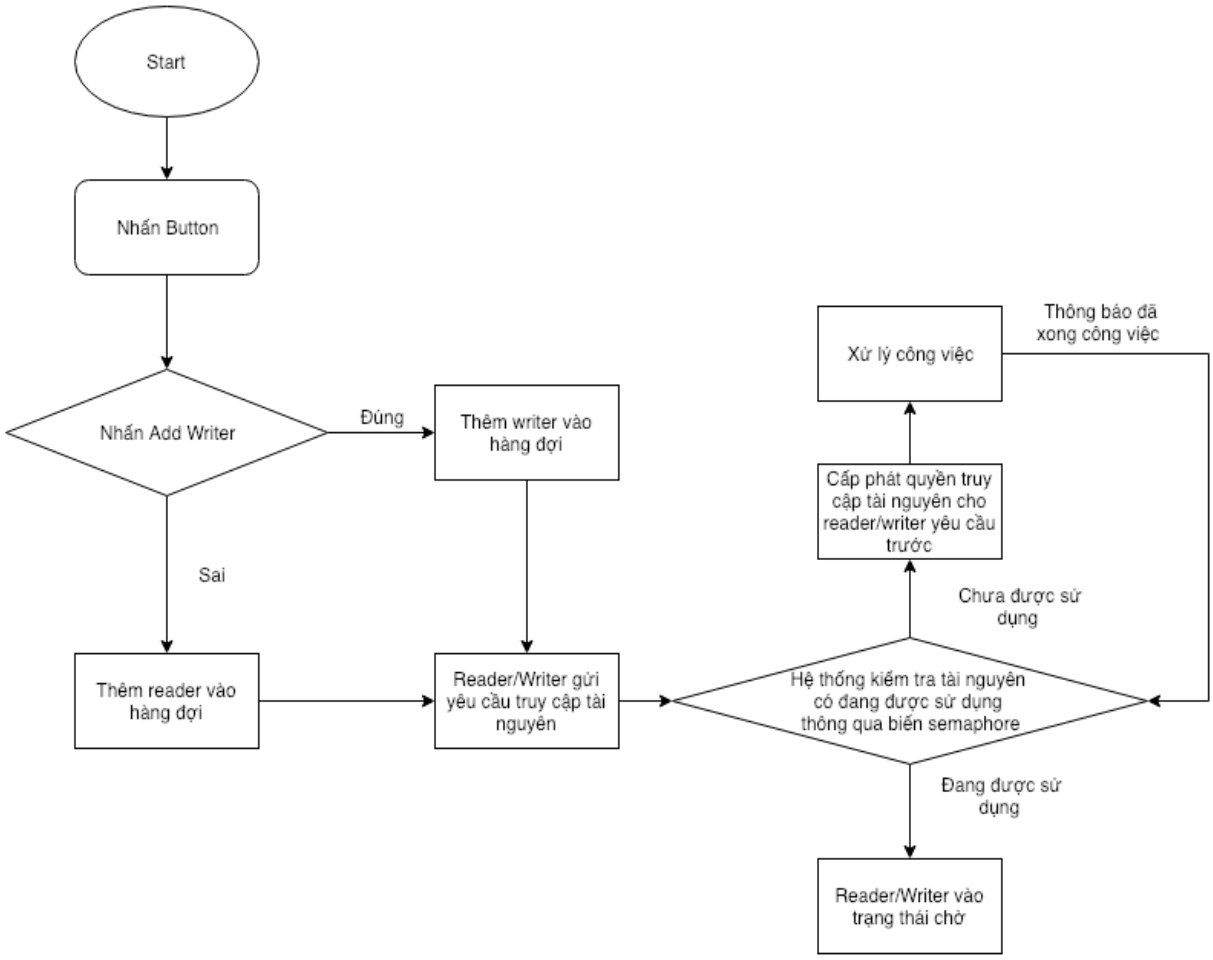
2. Xây dựng chương trình

- Các lớp được sử dụng trong chương trình:
  - Lớp Operator: Là lớp cha của lớp Reader/Writer. Thừa kế từ lớp JPanel, thể hiện một Writer/Reader trên màn hình.
  - Lớp Writer: Là lớp con của lớp Operator, implements Runnable interface. Lớp này chứa một biến reference đến tài nguyên được sử dụng chung, hai biến semaphore để điều khiển việc truy cập tài nguyên. Mỗi object của lớp writer có thể remove hoặc add thêm một item trong stack (Là tài nguyên sử dụng chung, chứa các item). Mỗi writer được thể hiện bằng một phần
  - Lớp Reader: Là lớp con của lớp Operator, implements Runnable interface. Lớp này chứa một biến reference đến tài nguyên được sử dụng chung, ba biến semaphore để điều khiển việc truy cập dữ liệu. Mỗi object của lớp reader có thể đọc số lượng item có trong stack (Là tài nguyên sử dụng chung, chứa các item) và in ra màn hình. Mỗi reader được thể hiện bằng một phần tử màu xanh trên màn hình.

- Lớp Item: Thừa kế từ lớp JPanel, thể hiện một phần tử trong tài nguyên được sử dụng chung. Thể hiện bằng những phần tử màu vàng trên màn hình.
- Lớp ProcessUI: Lớp này xử lý phần UI cho toàn bộ chương trình.
- Lớp Main: Tạo ra một instance của lớp ProcessUI để bắt đầu chương trình.

- Mô tả chương trình: Chương trình sẽ hiển thị hai button để người dùng tạo ra các reader, writer tương ứng. Với mỗi reader/writer được tạo ra, reader/writer đó được thể hiện bằng các phần tử xanh lá/đỏ tương ứng và được thêm vào vùng chờ trước khi được cấp phát quyền truy cập dữ liệu. Khi reader/writer được cấp phát quyền truy cập, chúng sẽ được chuyển sang vùng service (thể hiện reader/writer đang sử dụng tài nguyên). Sau khi sử dụng xong thì các reader/writer đó sẽ bị remove khỏi vùng service và quyền truy cập sẽ được cấp phát cho các reader/writer đang chờ trong khu vực chờ.

3. Sơ đồ hoạt động



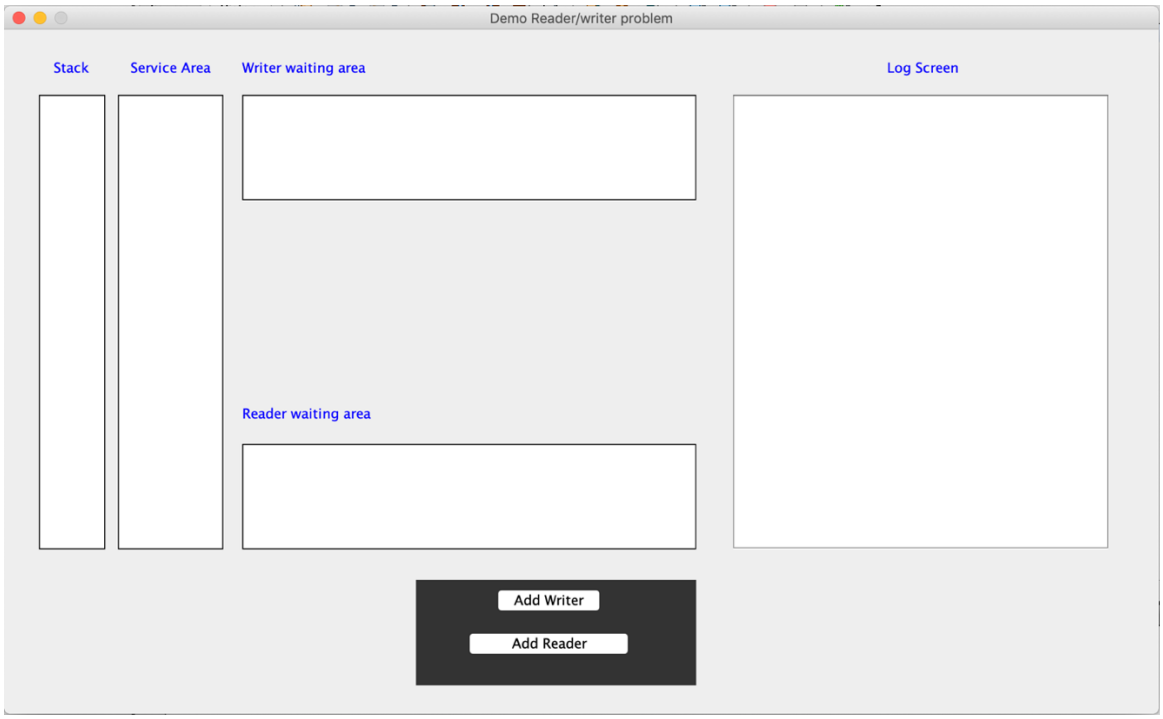
Hình 2: Sơ đồ hoạt động reader/writer

4. Môi trường phát triển

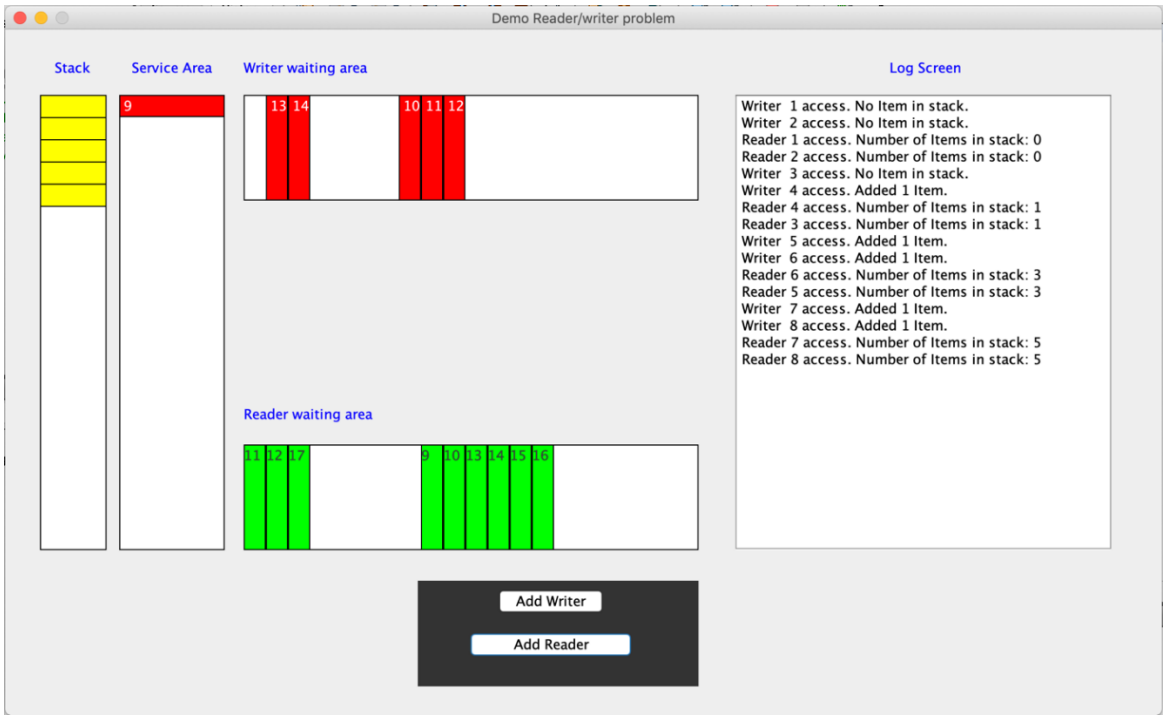
- Ngôn ngữ lập trình Java.
- IDE Netbeans
- Hệ điều hành Mac os

CHƯƠNG 3. TRIỂN KHAI VÀ ĐÁNH GIÁ CHƯƠNG TRÌNH

3.1 Kết quả demo chương trình



Hình 3: Chương trình khi chưa thêm reader/writer



Hình 4: Chương trình khi đang làm việc với reader/writer

3.2 Đánh giá kết quả chương trình

- Chương trình đã thực hiện được yêu cầu bài toán đặt ra. Các thread được cấp phát quyền sử dụng tài nguyên một cách hợp lý, không gây ra hiện tượng deadlock.

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

1. Kết luận

- Ưu điểm: chương trình đã thực hiện các yêu cầu bài toán đặt ra, có thể hiện qua giao diện trực quan.
- Hạn chế: Giao diện người dùng chưa đẹp.

2. Hướng phát triển

- Cần phải cho người dùng tương tác với dữ liệu vào của chương trình. Đồng thời phát triển giao diện giúp chương trình trực quan và thân thiện hơn.

PHẦN II: LẬP TRÌNH MẠNG
TIÊU ĐỀ: XÂY DỰNG TRÒ CHƠI CỜ VUA QUA MẠNG LAN

CHƯƠNG I. CƠ SỞ LÝ THUYẾT

1. Tổng quan về TCP/IP

1.1 Bộ giao thức liên mạng (IP Protocol)

- Giao thức liên mạng, thường gọi là giao thức IP (Internet Protocol) là một giao thức mạng hoạt động ở tầng 3 của mô hình OSI, nó qui định cách thức định địa chỉ các máy tính và cách thức chuyển tải các gói tin qua một liên mạng. IP được đặc tả trong bảng báo cáo kỹ thuật có tên RFC (Request For Comments) mã số 791 và là giao thức chủ yếu trong Bộ giao thức liên mạng. Cùng với giao thức TCP, IP trở thành trái tim của bộ giao thức Internet.
- IP có hai chức năng chính :
  - cung cấp dịch vụ truyền tải dạng không nối kết để chuyển tải các gói tin qua một liên mạng.
  - phân mãnh cũng như tập hợp lại các gói tin để hỗ trợ cho tầng liên kết dữ liệu với kích thước đơn vị truyền dữ liệu là khác nhau.
- Cấu trúc gói tin IP:

<table>
<tr><td>Version</td><td>IHL</td><td>Type of Service</td><td colspan="2">Total Length</td></tr>
<tr><td colspan="3">Identification</td><td>Flags</td><td>Fragment Offset</td></tr>
<tr><td colspan="2">Time to Live</td><td>Protocol</td><td colspan="2">Header Checksum</td></tr>
<tr><td colspan="5">Source IP Address</td></tr>
<tr><td colspan="5">Destination IP Address</td></tr>
<tr><td colspan="4">IP Options</td><td>Padding</td></tr>
<tr><td colspan="5">DATA</td></tr>
</table>

Hình 5: Cấu trúc gói tin IP

- **Version (Phiên bản):** Xác định phiên bản của giao thức đang được sử dụng.
- **IP Header Length (***Chiều dài của phần tiêu đề* : Xác định chiều dài của phần tiêu đề của gói tin, tính bằng đơn vị là từ - 32 bits (32-bit word).
- **Type-of-Service (Kiểu của dịch vụ :** Đặc tả mức độ quan trọng mà giao thức phía trên muốn xử lý gói tin.
- **Total Length (Tổng chiều dài gói tin):** Đặc tả chiều dài, tính bằng byte, của cả gói tin IP, bao gồm cả phần dữ liệu và tiêu đề.

- **Identification (Số nhận dạng):** Số nguyên nhận dạng gói tin dữ liệu hiện hành. Trường này được sử dụng để ráp lại các phân đoạn của gói tin.
- **Flags (Cờ hiệu):** Gồm 3 bít, bit có trọng số nhỏ để xác định gói tin có bị phân đoạn hay không. Bit thứ 2 xác định có phải đây là phân đoạn cuối cùng của gói tin hay không. Bit có trọng số lớn nhất chưa sử dụng.

- **Fragment Offset (Vị trí của phân đoạn):** Biểu thị vị trí của phân đoạn dữ liệu so với vị trí bắt đầu của gói dữ liệu gốc, nó cho phép máy nhận xây dựng lại gói tin ban đầu.
- **Time-to-Live (***Thời gian sống của gói tin)*: Lưu giữ bộ đếm thời gian, giá trị sẽ được giảm dần đến khi nó có giá trị là 0 thì gói tin sẽ bị xóa. Điều này giúp ngăn ngừa tình trạng gói tin được truyền đi lòng vòng không bao giờ đến được đích.
- **Protocol(Giao thức):** Biểu hiện giao thức ở tầng trên sẽ nhận gói tin khi nó đã được giao thức IP xử lý.
- **Header Checksum (Tổng kiểm tra lỗi tiêu đề)**: kiểm tra tính toàn vẹn của phần tiêu đề.
- S**ource Addres : Địa của máy gởi gói tin**.
- **Destination Address**: Địa chỉ của máy nhận gói tin.
- **Options**: Tùy chọn cho phép để hỗ trợ một số vấn đề, chẳng hạn vấn đề bảo mật.
- **Data**: Chứa dữ liệu của tầng trên gởi xuống cần truyền đi.

1.2 Bộ giao thức điều khiển giao vận (TCP Protocol)

- Giao thức TCP (Transmission Control Protocol - "Giao thức điều khiển truyền vận") là một trong các giao thức cốt lõi của bộ giao thức TCP/IP. Sử dụng TCP, các ứng dụng trên các máy chủ được nối mạng có thể tạo các "kết nối" với nhau, mà qua đó chúng có thể trao đổi dữ liệu hoặc các gói tin. Giao thức này đảm bảo chuyển giao dữ liệu tới nơi nhận một cách đáng tin cậy và đúng thứ tự. TCP còn phân biệt giữa dữ liệu của nhiều ứng dụng (chẳng hạn, dịch vụ Web và dịch vụ thư điện tử) đồng thời chạy trên cùng một máy chủ.
- TCP hỗ trợ nhiều giao thức ứng dụng phổ biến nhất trên Internet và các ứng dụng kết quả, trong đó có WWW, thư điện tử và Secure Shell.
- Trong bộ giao thức TCP/IP, TCP là tầng trung gian giữa giao thức IP bên dưới và một ứng dụng bên trên. Các ứng dụng thường cần các kết nối đáng tin cậy kiểu đường ống để liên lạc với nhau, trong khi đó, giao thức IP không cung cấp những dòng kiểu đó, mà chỉ cung cấp dịch vụ chuyển gói

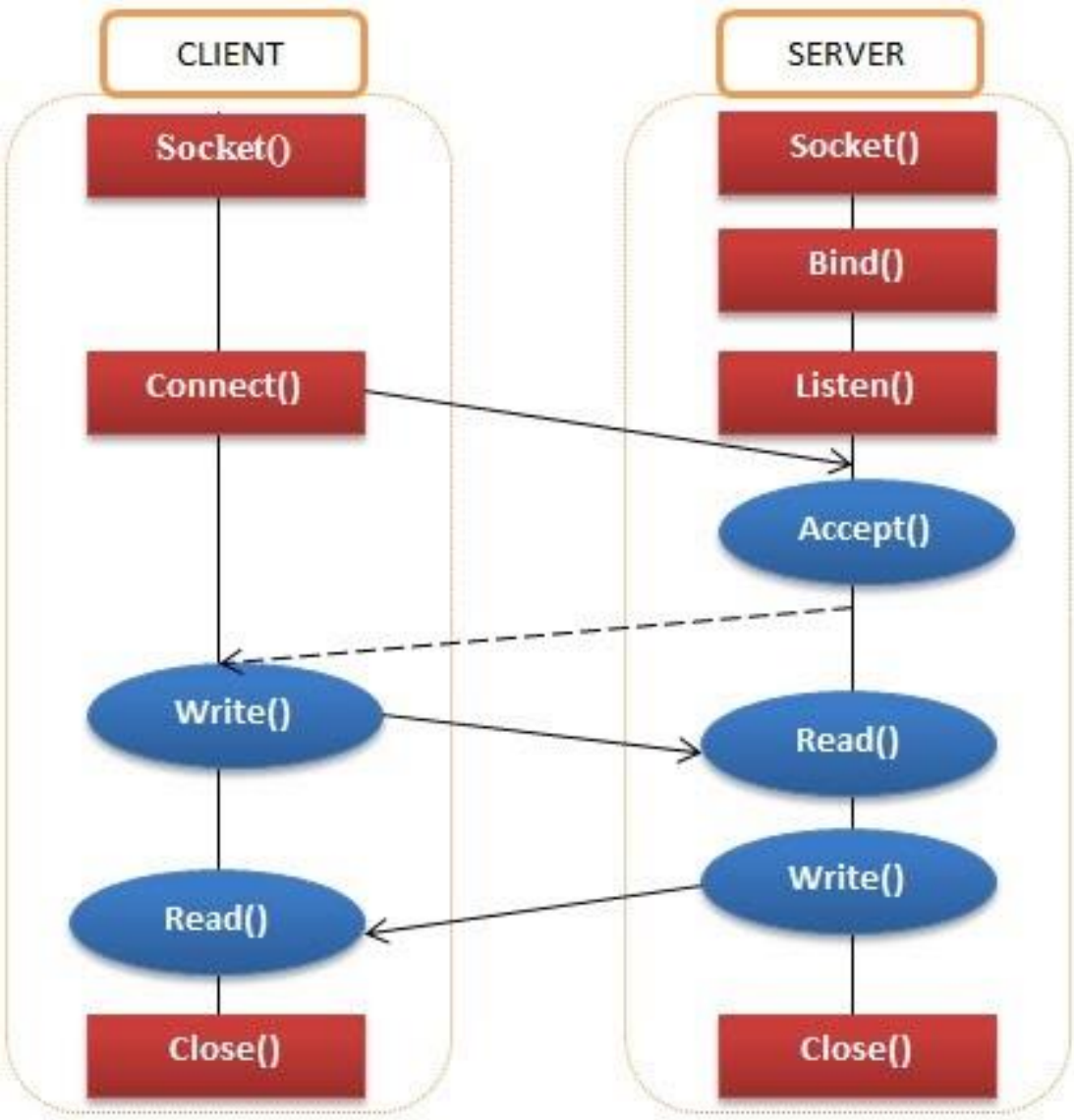
tin không đáng tin cậy. TCP làm nhiệm vụ của tầng giao vận trong mô hình OSI đơn giản của các mạng máy tính.

- Các ứng dụng gửi các dòng gồm các byte 8-bit tới TCP để chuyển qua mạng. TCP phân chia dòng byte này thành các đoạn (segment) có kích thước thích hợp (thường được quyết định dựa theo kích thước của đơn vị truyền dẫn tối đa (MTU) của tầng liên kết dữ liệu của mạng mà máy tính đang nằm trong đó). Sau đó, TCP chuyển các gói tin thu được tới giao thức IP để gửi nó qua một liên mạng tới mô đun TCP tại máy tính đích. TCP kiểm tra để đảm bảo không có gói tin nào bị thất lạc bằng cách gán cho mỗi gói tin một "số thứ tự" (sequence number). Số thứ tự này còn được sử dụng để đảm bảo dữ liệu được trao cho ứng dụng đích theo đúng thứ tự. Mô đun TCP tại đầu kia gửi lại "tin báo nhận" (acknowledgement) cho các gói tin đã nhận được thành công; một "đồng hồ" (timer) tại nơi gửi sẽ báo time-out nếu không nhận được tin báo nhận trong khoảng thời gian bằng một round-trip time (RTT), và dữ liệu (được coi là bị thất lạc) sẽ được gửi lại. TCP sử dụng checksum (giá trị kiểm tra) để xem có byte nào bị hỏng trong quá trình truyền hay không; giá trị này được tính toán cho mỗi khối dữ liệu tại nơi gửi trước khi nó được gửi, và được kiểm tra tại nơi nhận.

2. Lập trình Socket

2.1 Giới thiệu về Socket.

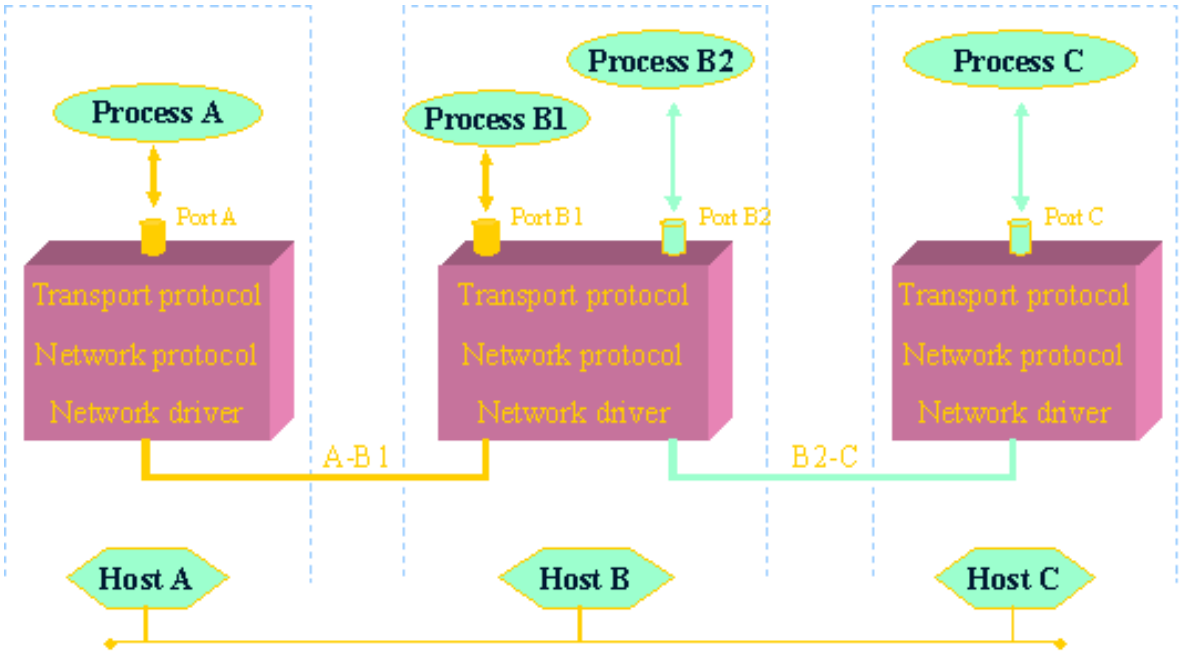
- Socket là một giao diện lập trình ứng dụng (API-Application Programming Interface). Socket cho phép thiết lập các kênh giao tiếp mà hai đầu kênh được đánh dấu bởi hai cổng (port). Thông qua các cổng này một quá trình có thể nhận và gởi dữ liệu với các quá trình khác.



Hình 6: Mô hình client - server

2.2 Số hiệu cổng của socket.

- Để có thể thực hiện các cuộc giao tiếp, một trong hai quá trình phải công bố số hiệu cổng của socket mà mình sử dụng. Mỗi cổng giao tiếp thể hiện một địa chỉ xác định trong hệ thống. Khi quá trình được gán một số hiệu cổng, nó có thể nhận dữ liệu gởi đến cổng này từ các quá trình khác. Quá trình còn lại cũng được yêu cầu tạo ra một socket.
- Ngoài số hiệu cổng, hai bên giao tiếp còn phải biết địa chỉ IP của nhau. Địa chỉ IP giúp phân biệt máy tính này với máy tính kia trên mạng TCP/IP. Trong khi số hiệu cổng dùng để phân biệt các quá trình khác nhau trên cùng một máy tính.

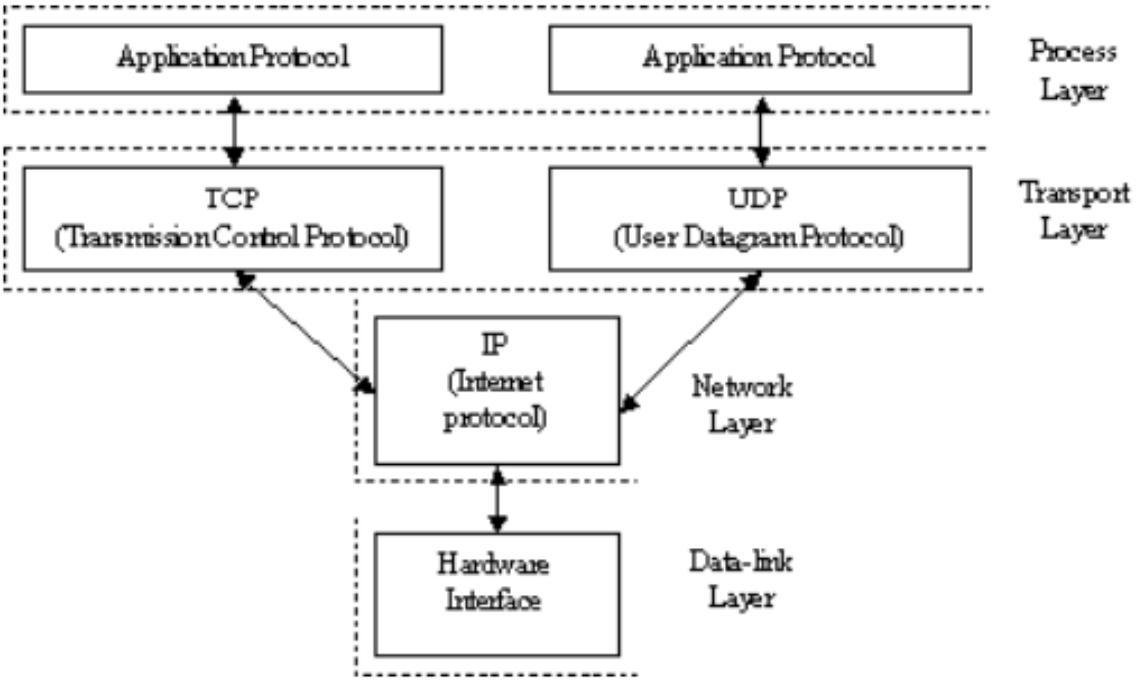


Hình 7: Số hiệu cổng

- Trong hình trên, địa chỉ của quá trình B1 được xác định bằng 2 thông tin: (Host B, Port B1):
- Địa chỉ máy tính có thể là địa chỉ IP dạng 203.162.36.149 hay là địa chỉ theo dạng tên miền như www.ukb.edu.vn
- Số hiệu cổng gán cho Socket phải duy nhất trên phạm vi máy tính đó, có giá trị trong khoảng từ 0 đến 65535 (16 bits). Trong đó, các cổng từ 1 đến 1023 được gọi là cổng hệ thống được dành riêng cho các quá trình của hệ thống.

2.3 Các chế độ giao tiếp.

- Xét kiến trúc của hệ thống mạng TCP/IP



Hình 8: Kiến trúc hệ thống TCP/IP

- Tầng vận chuyển giúp chuyển tiếp các thông điệp giữa các chương trình ứng dụng với nhau. Nó có thể hoạt động theo hai chế độ:
- Giao tiếp có nối kết, nếu sử dụng giao thức TCP
- Hoặc giao tiếp không nối kết, nếu sử dụng giao thức UDP
- Socket là giao diện giữa chương trình ứng dụng với tầng vận chuyển. Nó cho phép ta chọn giao thức sử dụng ở tầng vận chuyển là TCP hay UDP cho chương trình ứng dụng của mình.

CHƯƠNG II. PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

1. Yêu cầu bài toán

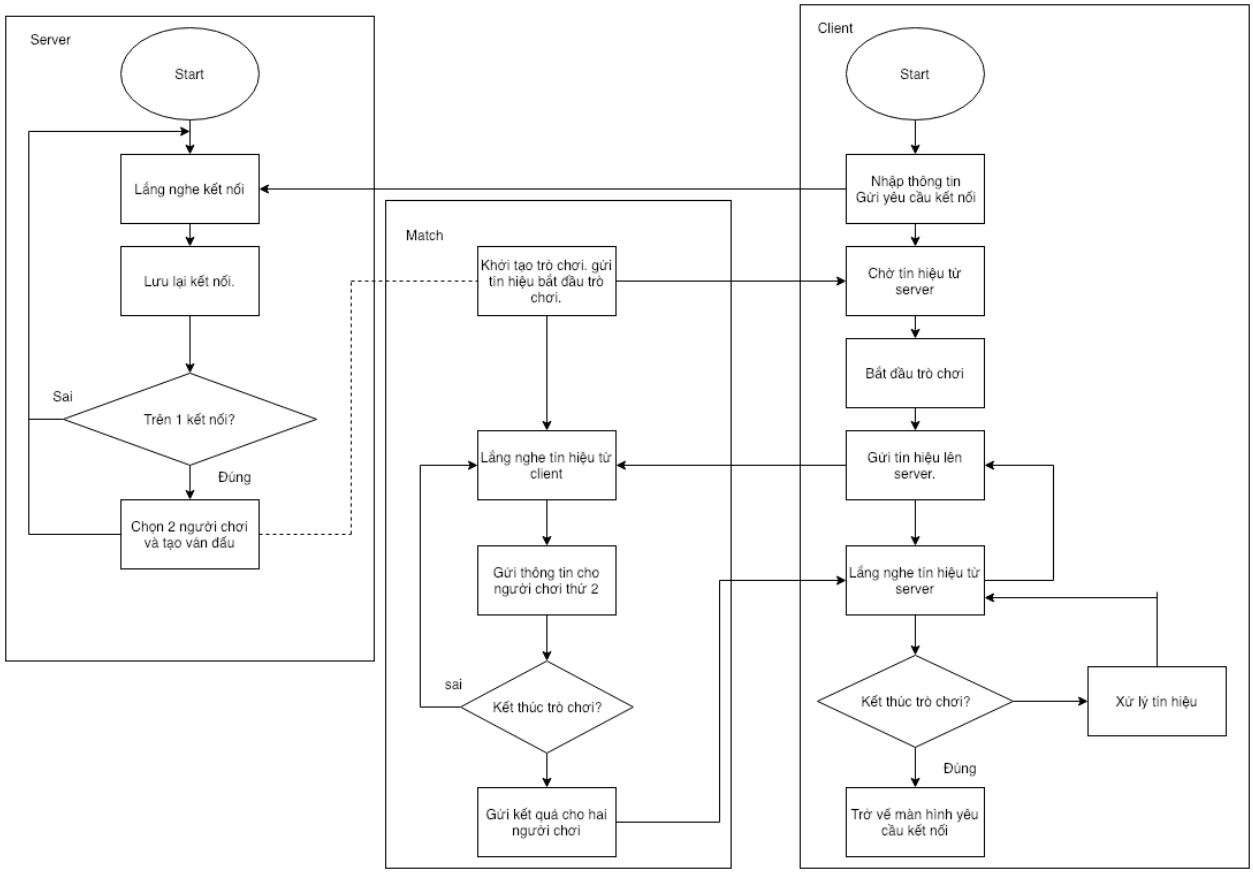
- Yêu cầu bài toán đặt ra là xây dựng một mô hình client – server giao tiếp với nhau thông qua Socket API để xử lý trò chơi cờ vua.
- Server được khởi đông sẽ lắng nghe các kết nối từ client. Khi có một kết nối từ client, server khởi tạo một kênh giao tiếp và lưu trữ kênh giao tiếp đó lại.
- Khi có từ 2 client trở lên, server khởi tạo một game đấu giữa 2 client. Server tiếp tục lắng nghe các kết nối từ client trong khi các game đấu diễn ra.
- Mỗi game đấu sẽ là một luồng xử lý riêng nhận thông tin giữa hai client trong lúc game đấu diễn ra.
- Các client sẽ thực hiện các nước đi và có thể trò chuyện qua lại thông qua kênh kết nối qua chức năng chat.
- Kết thúc một game đấu, server có trách nhiệm thông báo kết quả người chơi. Client có thể yêu cầu một kết nối mới để chơi tiếp một game khác.

2. Xây dựng chương trình

- Các class, enum, interface được sử dụng trong chương trình:
  - MessageType.java [Enum]: Thể hiện kiểu dữ liệu sẽ được gửi/nhận giữa client và server. Chương trình sẽ dựa vào kiểu dữ liệu này để xác định kiểu dữ liệu được nhận để xử lý một cách phù hợp.
  - Server.java [class]: Lớp này kế thừa lớp JFrame, sau khi nhận thông tin của port cần lắng nghe. Server sẽ tạo ra một luồng để lắng nghe tín hiệu kết nối từ client. Khi có từ 2 kết nối trở lên, server sẽ tạo ra một game đấu rồi tiếp tục lắng nghe kết nối từ client.
  - Player.java [class]: Lớp này thể hiện một người chơi, lưu trữ kết nối giữa server với client. Lớp Player kế thừa từ lớp Thread, lắng nghe và xử lý dữ liệu nhận từ client và báo về server.
  - Match.java [class]: Lớp này implement Runnable, PlayerInterface, thể hiện một game đấu. Lớp Match xử lý các sự kiện mà hai người chơi thực hiện, thông báo kết quả.
  - Client.java [class]: Lớp này kế thừa lớp JFrame và implement ChessboardInterface thể hiển một client. Mỗi client sẽ chịu trách nhiệm lấy thông tin người chơi, yêu cầu kết nối tới server và xử lý và gửi dữ liệu đến server.

- Chessboard.java [class]: Lớp này mô tả một bàn cờ, bao gồm các chức năng di chuyển quân cờ khi thực hiện một nước đánh, khởi tạo bàn cờ, hiển thị tin nhắn giữa hai người chơi,…
- Piece.java [class]: Lớp này là lớp asbtract, thể hiện một quân cờ, chứa các phương thức và thuộc tính chung của các quân cờ.
- King.java [class]: Lớp này thể hiện quân vua trong bàn cờ, kế thừa từ lớp Piece.
- Queen.java [class]: Lớp này thể hiện quân hậu trong bàn cờ, kế thừa từ lớp Piece.
- Rook.java [class]: Lớp này thể hiện quân xe trong bàn cờ, kế thừa từ lớp Piece.
- Bishop.java [class]: Lớp này thể hiện quân pháo trong bàn cờ, kế thừa từ lớp Piece.
- Knight.java [class]: Lớp này thể hiện quân mã trong bàn cờ, kế thừa từ lớp Piece.
- Pawn.java [class]: Lớp này thể hiện quân chốt trong bàn cờ, kế thừa từ lớp Piece.
- Square.java [class]: Lớp này thể hiện một ô trong bàn cờ vua, kế thừa từ lớp Jpanel.
- Move.java [class]: Lớp này thể hiện một bước di chuyển của một quân cờ. Bao gồm một điểm nguồn và một điểm đích đều là kiểu.
- ChessboardInterface.java [Interface]: Interface này bao gồm các phương thức với mục đích thông báo cho một observer hành động được thực hiện tại một chessboard.
- PlayerInterface.java [Interface]: Interface này bao gồm các phương thức với mục đích thông báo cho một observer hành động được thực hiện tại một player.

3. Sơ đồ hoạt động.



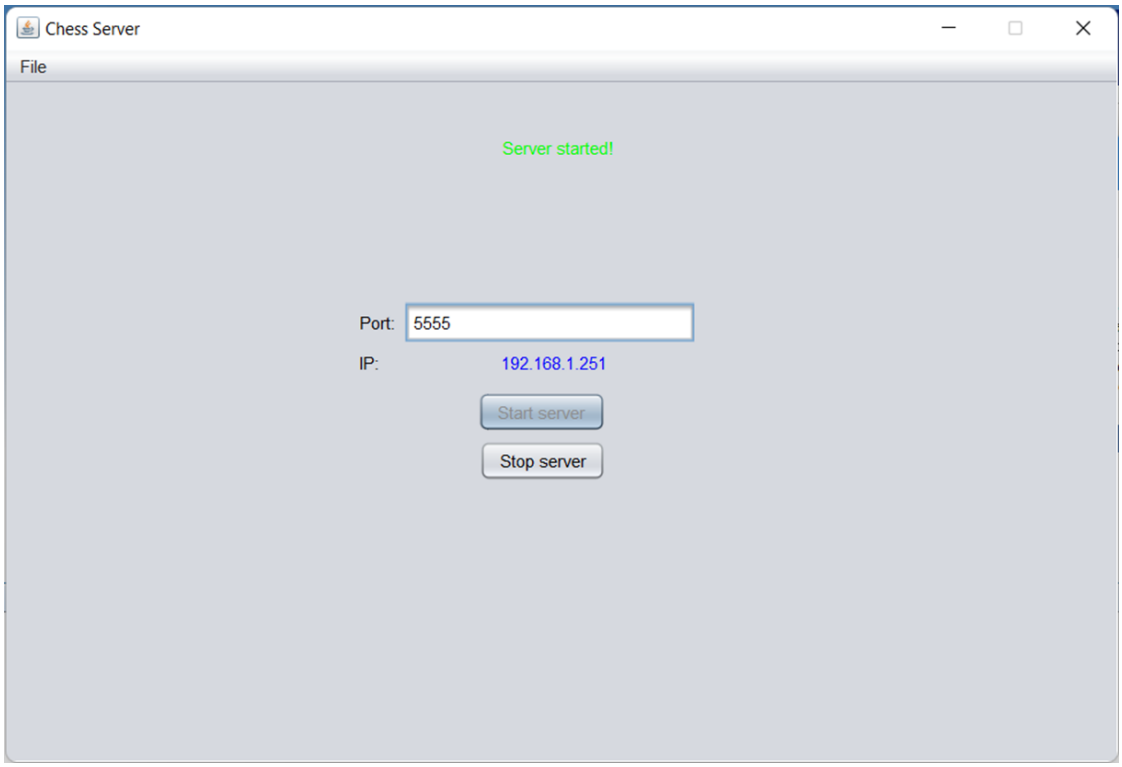
Hình 9: Sơ đồ hoạt động trò chơi cờ vua

4. Môi trường phát triển

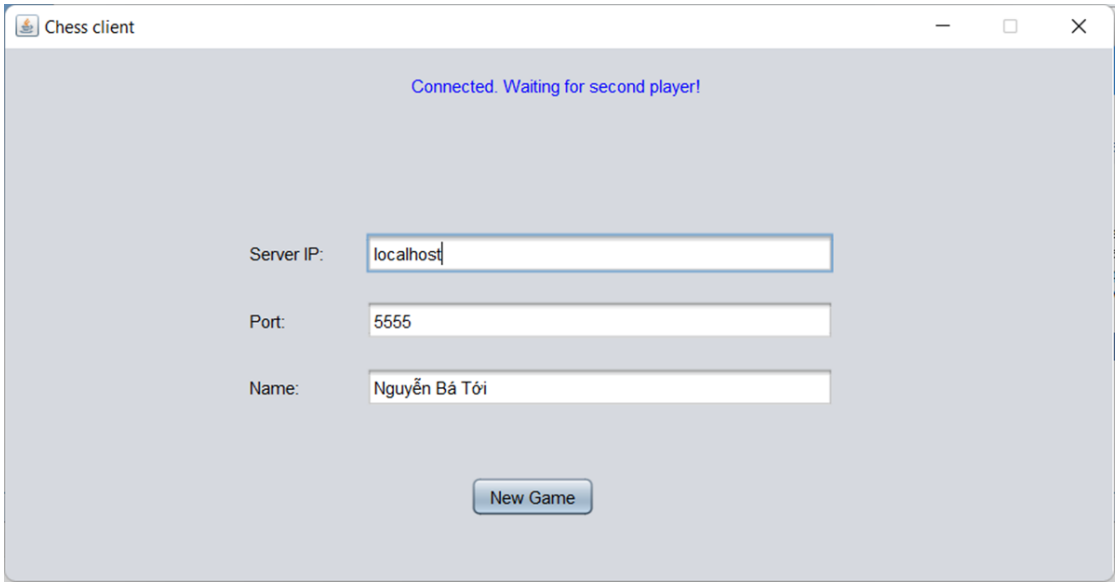
- Ngôn ngữ lập trình Java.
- IDE Netbeans
- Hệ điều hành Mac

CHƯƠNG III. TRIỂN KHAI VÀ ĐÁNH GIÁ KẾT QUẢ

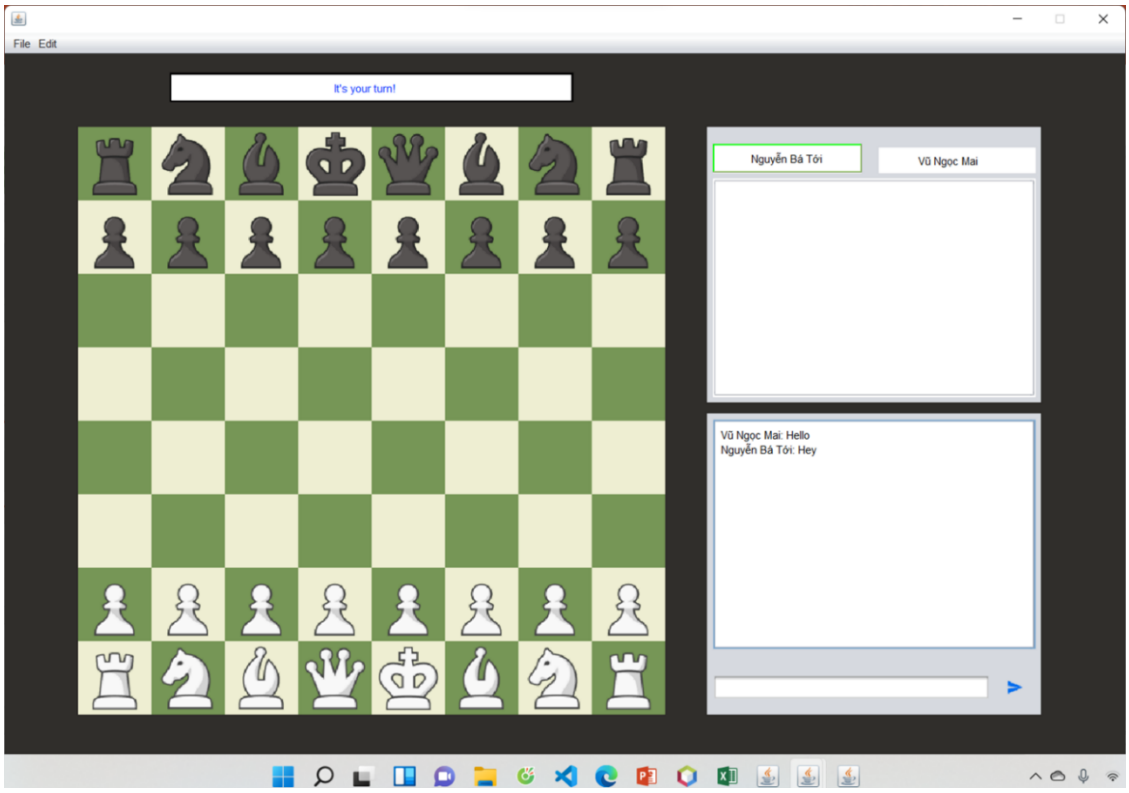
1. Kết quả demo chương trình



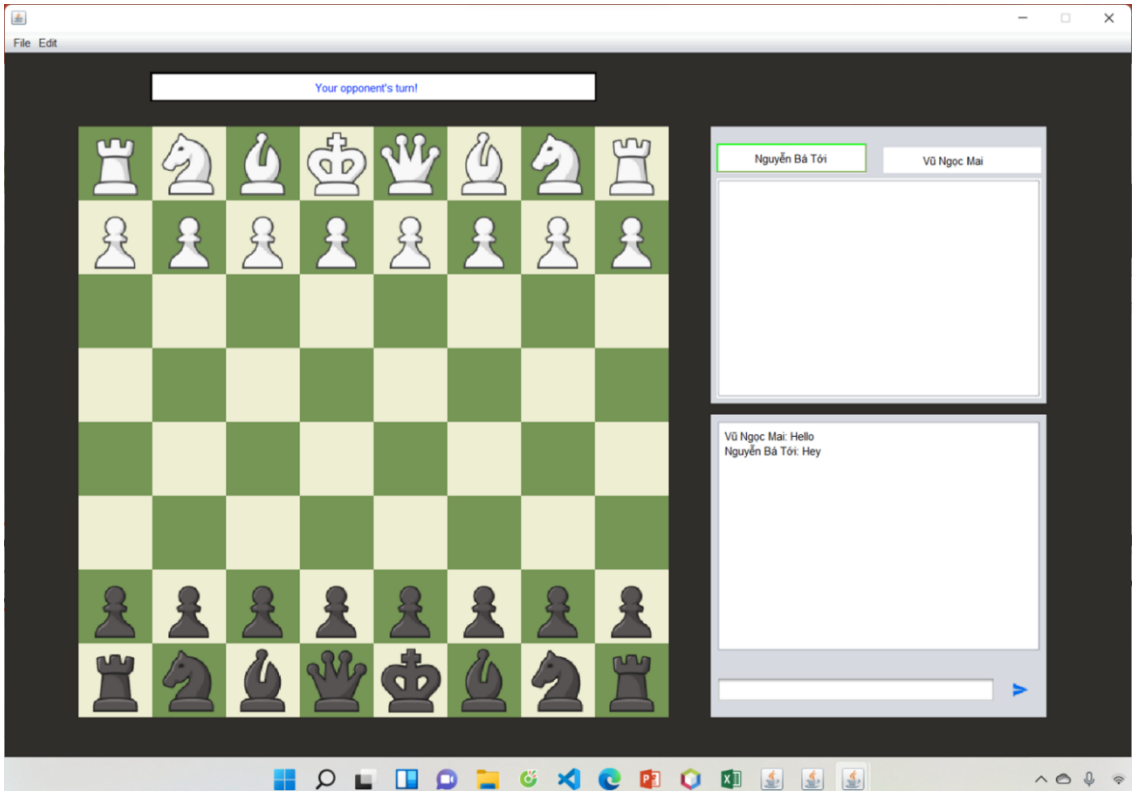
Hình 10: Server khởi động



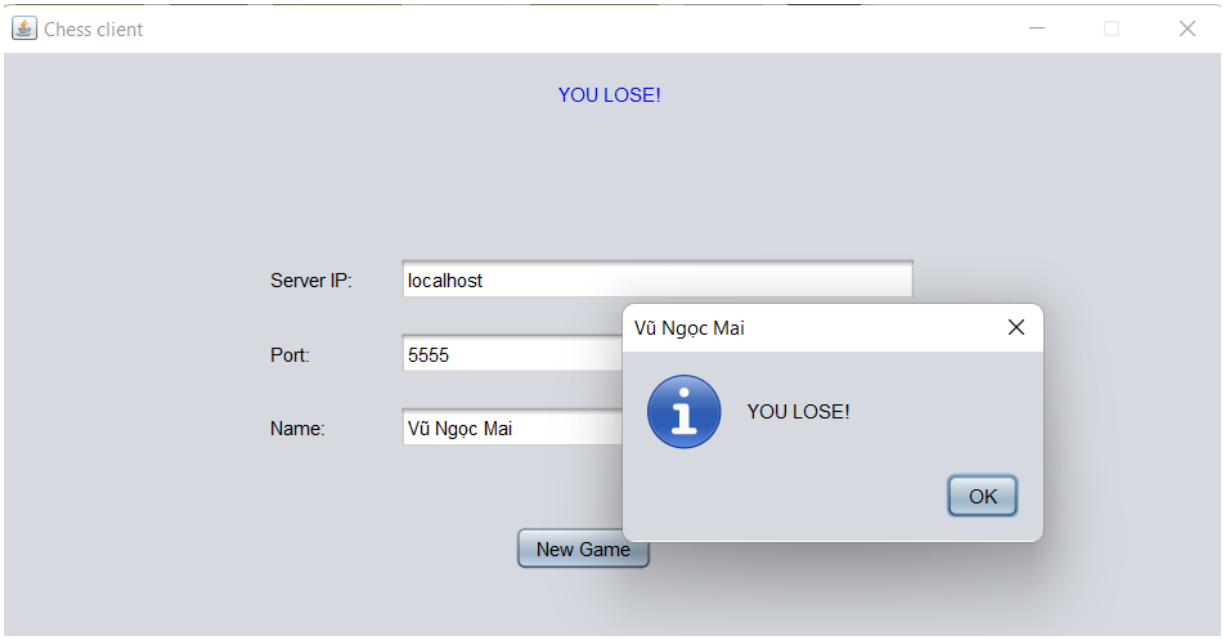
Hình 11: Nhập thông tin ở client



Hình 12: Bàn cờ mới bắt đầu trò chơi



Hình 13: Người chơi chat trong lúc chơi



Hình 14: Thông báo kết quả trò chơi

2. Đánh giá kết quả chương trình

- Chương trình xử lý được vấn đề đặt ra.
- Có thể tạo ra các game đấu môn cờ vua
- Xử lý được chức năng chat trong khi chơi

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

1. Kết luận

- Ưu điểm:
  - chương trình đã thực hiện các yêu cầu bài toán đặt ra
  - tạo ra giao diện trò chơi khá đẹp
  - người chơi có thể kết nối từ các máy tính khác nhau
- Hạn chế:
  - Chưa có chức năng chơi với bạn bè
  - luồng xử lý còn khá phức tạp và chồng chéo
  - Không lưu được kết quả người chơi

2. Hướng phát triển

- Bổ sung chức năng yêu cầu chơi với bạn bè
- Kết nối cơ sở dữ liệu để lưu kết quả người chơi
- Tối ưu hoá chương trình

KẾT LUẬN CHUNG

- Quá trình làm Bài tập lớn đã giúp chúng em có thêm nhiều hiểu biết hơn về Hệ Điều Hành và Lập Trình Mạng.
- Ngoài ra bài tập này cũng giúp chúng em hoàn thiện các kỹ năng của bản thân như: làm việc nhóm, quản lý thời gian, các viết và trình bày báo cáo, khả năng tự học, tìm hiểu những kiến thức chuyên ngành nhưng không được dạy trên trường…
- Lời cuối cùng nhóm em xin cảm ơn các thầy cô, các bạn đã tận tình chỉ dạy để chúng em có thể hoàn thành tốt bài tập này.

TÀI LIỆU THAM KHẢO

[1] Wikipedia: https://en.wikipedia.org/wiki/Readers%E2%80%93writers_problem

[2] voer.edu.vn: http://voer.edu.vn/c/bo-giao-thuc-lien-mang-ips-internet-protocols/b14d14a4/266af9cd

[3] chess.com: https://www.chess.com/play/computer

PHỤ LỤC