ĐẠI HỌC ĐÀ NẴNG
ĐẠI HỌC BÁCH KHOA
KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO ĐỒ ÁN
CƠ SỞ NGÀNH MẠNG

Phần nguyên lý hệ điều hành

Bài toán năm triết gia ăn tối

Phần lập trình mạng

Sử dụng Socket trong Java xây dựng ứng dụng quản lý danh bạ điện thoại theo mô hình Client_Server

Giảng viên : Ths.Trần Hồ Thủy Tiên
Sinh viên : Lê Xuân Mạnh
Lớp : 17T3
Mã số sinh viên : 102170170

Đà Nẵng 12-2020

MỤC LỤC

DANH MỤC HÌNH VẼ

LỜI MỞ ĐẦU

Trong ngành công nghệ thông tin nói chung, có rất nhiều môn học bổ ích mà sinh viên khám phá, học hỏi, trang bị những kiến thức phục vụ cho nhu cầu của lập trình viên sau này. Một trong số đó có Đồ án Cơ sở ngành Mạng, một môn học nền tảng mà tất cả sinh viên trong ngành thực hiện để có những kĩ năng tiếp cận với chuyên ngành của mình một cách dễ dàng hơn. Đồ án sẽ giúp sinh viên hiểu rõ hơn về lý thuyết môn học Nguyên lý hệ điều hành và Lập trình mạng đã được học. Có thể coi đồ án là một cơ hội, một dự án hay một bài tập lớn mà các thầy cô giáo dành cho sinh viên của mình.

Đối với đồ án lần này, hai lĩnh vực mà chúng em phải tìm hiểu và nghiên cứ hai đề tài tìm hiểu về bài toán năm triết gia ăn tối và sử dụng Socket trong Java xây dựng chương trình quản lý danh bạ theo mô hình Client-Server. Đây là một đồ án quan trọng giúp chúng em thống kê lại những kiến thức về hai môn học đã và đang hoàn thành, tìm hiểu sâu hơn và vận dụng để làm các chương trình mô phỏng.

Qua báo cáo này em xin chân thành cảm ơn thầy cô khoa Công nghệ Thông tin đã tạo điều kiện để chúng em có thể nghiên cứu kĩ hơn những kiến thức này, cách vận dụng vào thực tế và đặc biệt là cô Trần Hồ Thủy Tiên đã nhiệt tình theo dõi, hướng dẫn em trong quá trình thực hiện đề tài này.

Vì kiến thức còn hạn hẹp, nên không thể tránh khỏi những sai sót trong quá trình làm đề tài, rất mong nhận được sự góp ý của các thầy cô để sản phẩm được hoàn thiện hơn.

Em xin chân thành cảm ơn!

Sinh viên

Lê Xuân Mạnh

PHẦN I: NGUYÊN LÝ HỆ ĐIỀU HÀNH

TIÊU ĐỀ: **Bài toán năm triết gia ăn tối.**

Yêu cầu:

1. Mô tả bài toán.
2. Giới thiệu tài nguyên găng và đoạn găng:
3. Viết chương trình giải quyết bài toán 5 triết gia ăn tối. Chương trình tạo ra 5 tiến trình con mô phỏng hoạt động của 5 triết gia. Dùng semaphore để đồng bộ hoạt động của 5 triết gia này.

CHƯƠNG 1: CƠ SỞ LÝ THUYẾT

1.1. Tiến trình

1.1.1. Khái niệm

Theo wikipedia, trong khoa học máy tính, tiến trình là một thực thể điều khiển đoạn mã lệnh có một không gian địa chỉ, có ngăn xếp (stack) riêng lẽ, có bảng chứa các số miêu tả file(file desciptor) được mở cùng tiến trình và đặc biệt là có một định danh PID(process identifier) duy nhất trong toàn bộ hệ thống vào thời điểm tiến trình đang chạy.

Một số khái niệm khác của tiến trình, theo định nghĩa của Saltzer, tiến trình là một chương trình do một processor logic thực hiện, nhưng theo định nghĩa của Horning & Rendell thì tiến trình là một quá trình chuyển từ trạng thái này sang trạng thái khác dưới tác động của hàm hoạt động, xuất phát từ một trạng thái ban đầu nào đó.

1.1.2. Phân loại tiến trình

Các tiến trình trong hệ thống theo quan hệ về thời gian tồn tại có thể chia thành hai loại là:

- Tiến trình tuần tự là các tiến trình mà điểm khởi tạo của nó sau điểm kết thúc của tiến trình trước đó.
- Tiến trình song song (đồng thời) là các tiến trình mà điểm khởi tạo của tiến trình này nằm ở giữa các tiền trình khác, tức là có thể khởi tạo một tiến trình mới khi các tiến trình trước đó chưa kết thúc. Cũng có thể được chia thành nhiều loại như tiến trình song song độc lập, có quan hệ thông tin, phân cấp, đồng mức.

1.1.3. Luồng

Một luồng (Threads) là một đơn vị xử lý cơ bản trong hệ thống. Mỗi luồng xử lý tuần tự đoạn code của nó, sở hữu một con trỏ lệnh, tập các thanh ghi và một vùng nhớ

ngăn xếp riêng. Các luồng chia sẻ CPU với nhau như cách chia sẻ giữa các tiến trình, một luồng xử lý trong khi các luồng khác chờ đến lượt. Một luồng cũng có thể tạo lập ra các tiến trình con, và nhận các trạng thái khác nhau như một tiến trình thực sự. Một tiến trình có thể sở hữu nhiều luồng.

1.2. Tài nguyên găng và đoạn găng.

1.2.1. Tài nguyên găng (Critical Resourse)

Tài nguyên găng là tài nguyên mà tại mỗi thời điểm, chỉ có thể phục vụ được một tiến trình, khi có 2 hay nhiều tiến trình cùng có nhu cầu đối với tài nguyên thì tài nguyên đó trở thành tài nguyên găng.

Những tài nguyên găng thường phải chia sẻ cho nhiều tiến trình hoạt động đồng thời dùng chung, và có nguy cơ dễ dẫn đến sự tranh chấp giữa các tiến trình này khii sử dụng chúng. Tài nguyên găng có thể là tài nguyên phần cứng hoặc tài nguyên phần mềm, có thể là tài nguyên phân chia được hoặc không phân chia được, nhưng đa số thường là tài nguyên phân chia được như là: các biến dùng chung, các file chia sẻ.

Trong môi trường hệ điều hành đa nhiệm - đa chương – đa người sử dụng, việc chia sẻ tài nguyên cho các tiến trình của người sử dụng dùng chung là cần thiết, nhưng nếu hệ điều hành không tổ chức tốt việc sử dụng tài nguyên dùng chung của các tiến trình hoạt động đồng thời, thì không những không mang lại hiệu quả khai thác tài nguyên của hệ thống mà còn làm hỏng dữ liệu của các ứng dụng. Nguy hiểm hơn, việc hỏng dữ liệu này có thể hệ điều hành và ứng dụng không thể phát hiện được. Việc hỏng dữ liệu của ứng dụng có thể làm sai lệch ý nghĩa thiết kế của nó. Đây là điều mà cả hệ điều hành và người lập trình đều không mong muốn.

Các tiến trình hoạt động đồng thời thường cạnh tranh với nhau trong việc sử dụng tài nguyên dùng chung. Hai tiến trình hoạt động đồng thời cùng ghi vào một không gian nhớ chung (một biến chung) trên bộ nhớ hay hai tiến trình đồng thời cùng ghi dữ liệu vào một file chia sẻ, đó là những biểu hiện của sự cạnh tranh về việc sử dụng tài nguyên dùng chung của các tiến trình. Để các tiến trình hoạt động đồng thời không cạnh tranh hay xung đột với nhau khi sử dụng tài nguyên dùng chung hệ điều hành phải tổ chức cho các tiến trình này được độc quyền truy xuất/ sử dụng trên các tài nguyên dùng chung này.

1.2.2. Đoạn găng (Critical Section)

Đoạn chương trình (code) trong các tiến trình đồng thời, có chứa các tài nguyên găng được gọi là đoạn găng hay là miền găng.

Để hạn chế các lỗi có thể xảy ra do sử dụng tài nguyên găng, HĐH phải điều khiển các tiến trình sao cho, tại một thời điểm chỉ có một tiến trình nằm trong đoạn găng, nếu có nhiều tiến trình cùng muốn vào (thực hiện) đoạn găng thì chỉ có một tiến trình được vào, các tiến trình khác phải chờ, một tiến trình khi ra khỏi (kết thúc) đoạn găng phải báo cho hệ điều hành hoặc các tiến trình khác biết để các tiến trình này vào

đoạn găng,… Công tác của hệ điều hành điều khiển tiến trình thực hiện đoạn găng được gọi là đồng bộ hóa tiến trình hay điều độ tiến trình qua đoạn găng.

Việc điều độ tiến trình phải đạt được các yêu cầu sau:

1. Tại một thời điểm chỉ có một tiến trình được nằm trong đoạn găng.
2. Nếu có nhiều tiến trình đồng thời cùng yêu cầu vào đoạn găng thì chỉ có một tiến trình được phép vào đoạn găng, các tiến trình khác phải xếp hàng chờ trong hàng đợi.
3. Không cho phép một tiến trình nằm vô hạn trong đoạn găng và không cho phép một tiến trình chờ vô hạn trước đoạn găng (chờ trong hàng đợi).
4. Nếu tài nguyên găng được giải phóng thì hệ điều hành có nhiệm vụ đánh thức các tiến trình trong hàng đợi ra để tạo điều kiện cho nó vào đoạn găng.

Ngyên lý cơ bản của điều độ là tổ chức truy xuất độc quyền trên tài nguyên găng, nhưng sự bắt buộc độc quyền này còn tồn tại hai hạn chế lớn là có thể dẫn đến bế tắc(Deadlock) trong hệ thống và có thể các tiến trình bị đói tài nguyên (Starvation).

1.3. Giải pháp Semaphore

1.3.1. Định nghĩa

Giải pháp Semaphore (Đèn báo) này được Dijkstra đề xuất vào năm 1965. Semaphore được định nghĩa để sử dụng trong các sơ đồ điều độ như sau:

- Semaphore S là một biến nguyên, khởi gán bằng một giá trị không âm, đó là khả năng phục vụ của tài nguyên găng tương ứng với nó.
- Ứng với S có một hàng đợi F(s) để lưu các tiến trình đang bị blocked trên S.
- Chỉ có hai thao tác Down và Up được tác động đến semaphore S. Down giảm S xuống một đơn vị, Up tăng S lên một đơn vị. Mỗi tiến trình trước khi vào đoạn găng thì phải gọi Down để kiểm tra và xác lập quyền vào đoạn găng.
- Mỗi tiến trình ngay sau khi ra khỏi đoạn găng phải gọi Up để kiểm tra xem có tiến trình nào đang đợi trong hàng đợi hay không, nếu có thì đưa tiến trình trong hàng đợi vào đoạn găng.

1.3.2. Ý nghĩa

- Mỗi tiến trình chỉ kiểm tra quyền vào đoạn găng một lần, khi chờ nó không làm gì cả, tiến trình ra khỏi đoạn găng phải đánh thức nó.
- Không xuất hiện hiện tượng chờ đợi tích cực, nên khai thác tối đa thời gian xử lý của processor.
- Nhờ cơ chế hàng đợi mà hệ điều hành có thể thực hiện gán độ ưu tiên cho các tiến trình khi chúng ở trong hành đợi.
- Trị tuyệt đối của S cho biết số lượng các tiến trình đang đợi trên F(S).

1.4. Deadlock

1.4.1. Khái niệm

Deadlock (bế tắc) là hiện tượng trong hệ thống có hai hay nhiều tiến trình cùng chờ đợi một sự kiện, mà nếu sự kiện đó không xảy ra thì sự chờ đợi ấy là chờ vô hạn. Sự kiện thường là chờ được cấp tài nguyên, và sự chờ đợi này có thể kéo dài vô hạn nếu không có sự tác động từ bên ngoài.

1.4.2. Những điều kiện hay gây ra deadlock.

Năm 1971, Coffman đã đưa ra và chứng tỏ được rằng, nếu hệ thống tồn tại đồng thời bốn điều kiện sau đây thì hệ thống sẽ xảy ra tắc nghẽn:

- Tồn tại tài nguyên găng (điều kiện độc quyền): sử dụng tài nguyên này là loại trừ lẫn nhau (mutual exclusion) hay độc quyền sử dụng: Đối với các tài nguyên không phân chia được thì tại mỗi thời điểm chỉ có một tiến trình sử dụng được tài nguyên.
- Giữ và đợi (hold and wait): một tiến trình hiện tại đang chiếm giữ tài nguyên, lại xin cấp phát thêm tài nguyên mới.
- Đợi vòng tròn (Circular wait): mỗi tiến trình đang chiếm giữ tài nguyên mà tiến trình khác đang cần.
- Không có sự phân phối lại tài nguyên, không có tài nguyên nào có thể được giải phóng từ một tiến trình đang chiếm giữ nó.

Trong nhiều trường hợp các điều kiện thứ nhất và thứ hai là rất cần thiết đối với hệ thống. Sự thực hiện độc quyền là cần thiết để bảo đảm tính đúng đắn của kết quả và tính toàn vẹn của dữ liệu. Tương tự, sự ưu tiên không thể thực hiện một cách tuỳ tiện, đặt biệt đối với các tài nguyên có liên quan với nhau, việc giải phóng từ một tiến trình này có thể ảnh hưởng đên kết quả xử lý của các tiến trình khác.

Ba điều kiện đầu là điều kiện cần chứ không phải là điều kiện đủ để xảy ra tắc nghẽn. Điều kiện thứ tư là kết quả tất yếu từ ba điều kiện đầu.

1.4.3. Các phương pháp xử lý deadlock

Phần lớn, chúng ta có thể giải quyết vấn đề deadlock theo một trong ba cách:

- Chúng ta có thể sử dụng một giao thức để ngăn chặn hay tránh deadlocks, đảm bảo rằng hệ thống sẽ không bao giờ đi vào trạng thái deadlock.
- Chúng ta có thể cho phép hệ thống đi vào trạng thái deadlock, phát hiện nó và phục hồi.
- Chúng ta có thể bỏ qua hoàn toàn vấn đề này và giả vờ deadlock không bao giờ xảy ra trong hệ thống. Giải pháp này được dùng trong nhiều hệ điều hành, kể cả UNIX.

1.4.4. Ngăn chặn deadlock

Ngăn chặn bế tắc là thiết kế một hệ thống sao cho không để hiện tượng bế tắc xảy ra. Các phương thức phòng chống bế tắc đều tập trung giải quyết bốn điều kiện gây ra tắc nghẽn, sao cho hệ thống không thể xảy ra đồng thời bốn điều kiện trên.

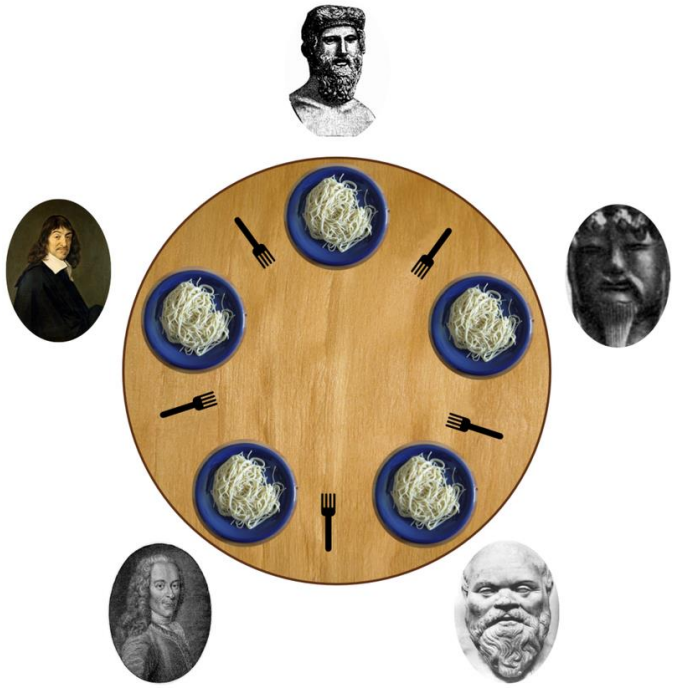
- Đối với điều kiện tồn tại tài nguyên găng (điều kiện độc quyền): Điều kiện này gần như không tránh khỏi, vì sự độc quyền là cần thiết đối với tài nguyên thuộc loại phân chia được như các biến chung, các tập tin chia sẻ, hệ điều hành cần phải hỗ trợ sự độc quyền trên các tài nguyên này. Tuy nhiên, với những tài nguyên thuộc loại không phân chia được hệ điều hành có thể sử dụng kỹ thuật SPOOL (Simultaneous Peripheral Operation Online) để tạo ra nhiều tài nguyên ảo cung cấp cho các tiến trình đồng thời.
- Đối với điều kiện giữ và đợi: Điều kiện này có thể ngăn chặn bằng cách buộc tiến trình yêu cầu tất cả tài nguyên mà nó cần tại một thời điểm và tiến trình sẽ bị khoá (blocked) cho đến khi yêu cầu tài nguyên của nó được hệ điều hành đáp ứng. Phương pháp này không hiệu quả. Thứ nhất, tiến trình phải đợi trong một khoảng thời gian dài để có đủ tài nguyên mới có thể chuyển sang hoạt động được, trong khi tiến trình chỉ cần một số ít tài nguyên trong số đó là có thể hoạt động được, sau đó yêu cầu tiếp. Thứ hai, lãng phí tài nguyên, vì có thể tiến trình giữ nhiều tài nguyên mà chỉ đến khi sắp kết thúc tiến trình mới sử dụng, và có thể đây là những tài nguyên mà các tiến trình khác đang rất cần. Ở đây hệ điều hành có thể tổ chức phân lớp tài nguyên hệ thống. Theo đó tiến trình phải trả tài nguyên ở mức thấp mới được cấp phát tài nguyên ở cấp cao hơn.
- Đối với điều kiện chờ đợi vòng tròn: Điều kiện này có thể ngăn chặn bằng cách phân lớp tài nguyên của hệ thống. Theo đó, nếu một tiến trình được cấp phát tài nguyên ở ớp L, thì sau đó nó chỉ có thể yêu cầu các tài nguyên ở lớp thấp hơn lớp L.
- Đối với điều kiện phân phối lại tài nguyên: Điều kiện này có thể ngăn chặn bằng cách, khi tiến trình bị rơi vào trạng thái khoá, HĐH có thể thu hồi tài nguyên của tiến trình bị khoá để cấp phát cho tiến trình khác và cấp lại đầy đủ tài nguyên cho tiến trình khi tiến trình được đưa ra khỏi trạng thái khoá.

CHƯƠNG 2: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

2.1. Yêu cầu bài toán

Tên đề tài là giải quyết bài toán "Năm triết gia ăn tối". Mục đích của bài toán là giải quyết vấn đề " đồng bộ giữa các tiến trình", các vấn đề tắc nghẽn có thể xảy ra. Mục tiêu của đề tài là viết chương trình giải quyết bài toán "Năm triết gia ăn tối". Chương trình phải tạo ra năm quá trình con mô phỏng như hoạt động của năm triết gia. Yêu cầu sử dụng giải pháp Semaphore đề thực hiện đồng bộ giữa năm triết gia.

Bài toán được đưa ra năm 1965 bởi tác giả Edsger Dijkstra. Bài toán được mô tả như sau:



Hình 1. Hình ảnh mình họa cho bài toán năm triết gia ăn tối

Có năm nhà triết gia, vừa suy nghĩ vừa ăn tối. Các triết gia ngồi trên một bàn tròn, trước mặt họ là các đĩa thức ăn, mỗi người một địa. Có 5 chiếc nĩa được đặt xe kẽ giữa các triết gia như hình trên. Khi một triết gia suy nghĩ, ông ta không giao tiếp với các triết gia khác. Thỉnh thoảng, một triết gia cảm giác thấy đói và cố gắng chọn hai chiếc nĩa gần nhất, là hai chiếc nĩa nằm giữa ông ta và hai triết gia gần nhất bên trái và phải. Một triết gia chỉ lấy được một chiếc nĩa tại một thời điểm. Chú ý, ông ta không thể lấy chiếc nĩa mà nó đang được dùng với triết gia khác bên cạnh. Khi một triết gia đói và có hai chiếc nĩa cùng một lúc, ông ta ăn mà không đặt nĩa xuống. Khi triết gia ăn xong, ông ta đặt cái nĩa xuống và bắt đầu suy nghĩ tiếp. Một triết gia có thể bị chết đói nếu ông ta không có cách nào để ăn được. Vấn đề bài toán là giải quyết tình trạng tắc nghẽn do các triết gia phải đợi lẫn nhau nên không có ai ăn được.

2.2. Thuật toán sử dụng

Sử dụng giải pháp Semaphore đề thực hiện đồng bộ giữa năm triết gia.

2.3. Thiết kế hệ thống

Trong bài toán Năm triết gia ăn tối, xem mỗi triết gia là một tiến trình, những chiếc nĩa (Fork) là tài nguyên dùng chung (tài nguyên găng) cần được bảo vệ. Chương trình xây dựng sẽ giải quyết hai vấn đề chính, đó là Quản lý vùng găng và Ngăn chặn Deadlock.

2.3.1. Quản lý vùng găng

Một khối đồng bộ (Synchronized) block đánh dấu một phương thức hay một khối mã là được đồng bộ. Sử dụng khối đồng bộ trong Java có thể tránh xảy ra xung đột.

Biến taken được xây dựng trong class Fork để quản lý tài nguyên dùng chung (fork), mỗi fork được tạo ra sẽ có 1 biến khóa taken kiểu boolean để đánh dấu:

- taken = true: nĩa tương ứng đang được sử dụng.

- taken = false: nĩa tương ứng chưa được sử dụng.

Việc đóng/mở vùng găng được thể hiện qua hai phương thức synchronized put() và get() trong class Fork:

```
//Phương thức put
synchronized void put() {
    taken = false;
    display.setFork(identity, taken);
    notify();
}

//Phương thức get
synchronized void get() throws InterruptedException {
    while (taken)
        wait();
    taken = true;
    display.setFork(identity, taken);
}
```

Hình 2. Hàm put và get trong Class Fork.

Phương thức put() tương ứng với hành động một triết gia đặt chiếc nĩa xuống khi ăn xong. Thuộc tính taken được gián bằng false, tức là chiếc nĩa trở lại trạng thái như lúc chưa có ai sử dụng để các triết gia khác có thể tìm thấy và sử dụng tài nguyên.

Ngược lại, phương thức get() tương ứng với hành động một triết gia lấy một chiếc nĩa khi thấy đói. Trong phương thức này, ta cần kiểm tra trạng thái của chiếc nĩa trước khi cho phép các triết gia lấy nó. Nếu thuộc tính taken đang có giá trị bằng true, tức chiếc nĩa đang được sử dụng) thì ta phải đợi cho đến khi biến taken bằng false (có một triết gia đã đặt chiếc nĩa xuống). Sau khi triết gia lấy chiếc nĩa thì gán biến taken bằng true để đánh dấu chiếc nĩa đã được sử dụng.

Như vậy, class Fork đã giải quyết được vấn đề tài nguyên dùng chung trong bài toán thông qua hai hàm của hệ thống wait() và notify(). Nói cách khác hàm wait() cho phép tiến trình được loại ra khỏi danh sách đang hoạt động cho đến khi một tiến trình

khác gọi hàm notify(), hàm này thông báo và kích hoạt trở lại tiến trình đầu tiên gọi wait() trên cùng một đối tượng.

2.3.2. Giải pháp ngăn chặn deadlock.

Vấn đề deadlock có thể tránh khỏi bằng cách xây dựng hai phương thức Wait() và Signal() trong class Philosopher (đại diện cho triết gia) sao cho vòng tròn đợi không xảy ra. Các phương thức được cài đặt như sau:

```
public void Wait() throws InterruptedException {
    if (identity % 2 == 0) {
        controller.setPhil(identity, controller.HUNGRY);
        hungry();
        right.get();
        // gotright fork
        controller.setPhil(identity, controller.GOTRIGHT);
        sleep(500);
        left.get();
    } else {
        controller.setPhil(identity, controller.HUNGRY);
        hungry();
        left.get();
        // gotleft fork
        controller.setPhil(identity, controller.GOTLEFT);
        sleep(500);
        right.get();
    }
}

public void Signal() {
    right.put();
    left.put();
}
```

Hình 3. Các phương thức để giải quyết Deadlock

Cách giải quyết ở đây là những triết gia với số thứ tự chẵn khi đói sẽ ưu tiến lấy nĩa bên phải của mình trước, sau đó đến nĩa bên trái, những triết gia có số thứ tự lẻ thì ngược lại. Như vậy hai triết gia cạnh nhau sẽ có thứ tự lấy nĩa khác nhau, tránh được vòng tròn đợi, bởi vì nếu mỗi triết gia đều lấy nĩa bên trái trước, hoặc bên phải trước thì sẽ có nguy cơ dẫn đến vòng tròn đợi. Đây còn gọi là giải pháp bất đối xứng.

2.3.3. Cấu trúc chương trình

Chương trình được xây dựng gồm 3 class:

- Class Fork: đại diện cho nĩa (tài nguyên găng).
- Class Philosopher: đại diện cho triến gia (tiến trình).
- Class DiningPhilosophers: Khởi tạo các class khác và quản lý giao diện.

CHƯƠNG 3: TRIỂN KHAI VÀ ĐÁNH GIÁ KẾT QUẢ

3.1. Môi trường triển khai

- Hệ điều hành windows 10.
- Chương trình được viết bằng ngôn ngữ Java.
- Phần mềm Eclipse.

3.2. Triển khai

3.2.1. Lớp Philosopher

Class gồm các thuộc tính:

```
private int identity;//Đánh số thức tự của các triết gia
private DiningPhilosophers controller;//Dùng để cập nhật giao diện
private Fork left;//Nĩa trái
private Fork right;//Nĩa phải
private JLabel philLable;
private JLabel comments;
private JLabel meals;
private JLabel forkLeft, forkRight;
private Random randomPeriod = new Random();
int numberOfMeals = 0;
```

Vì class Philosopher đại diện cho tiến trình nên nó kế thừa class Thread trong Java, và vì là một Thread nên những tác vụ của nó được đưa vào phương thức run():

```
@Override
public void run() {
    super.run();
    try {
        while (true) {
            // thinking
            controller.setPhil(identity, controller.THINKING);
            think();
            sleep(50 * randomPeriod.nextInt(100));
            // hungry
            Wait();
            // eating
            controller.setPhil(identity, controller.EATING);
            System.out.println("Philosopher# " + identity + " Eating");
            eat();
            sleep(50 * randomPeriod.nextInt(100));
            Signal();
            controller.Message += "   Philosopher " + identity + " EATED" + "\n";
            controller.txtMes.setText(controller.Message);
        }
    } catch (InterruptedException e) {
        // TODO: handle exception
    }
}
```

Phương thức run() mô tả trực quan những hành động luân phiên và lặp lại của triết gia, đó là: suy nghĩ – đói – ăn.

3.2.2. Lớp Fork

Class Chopstick gồm các thuộc tính:

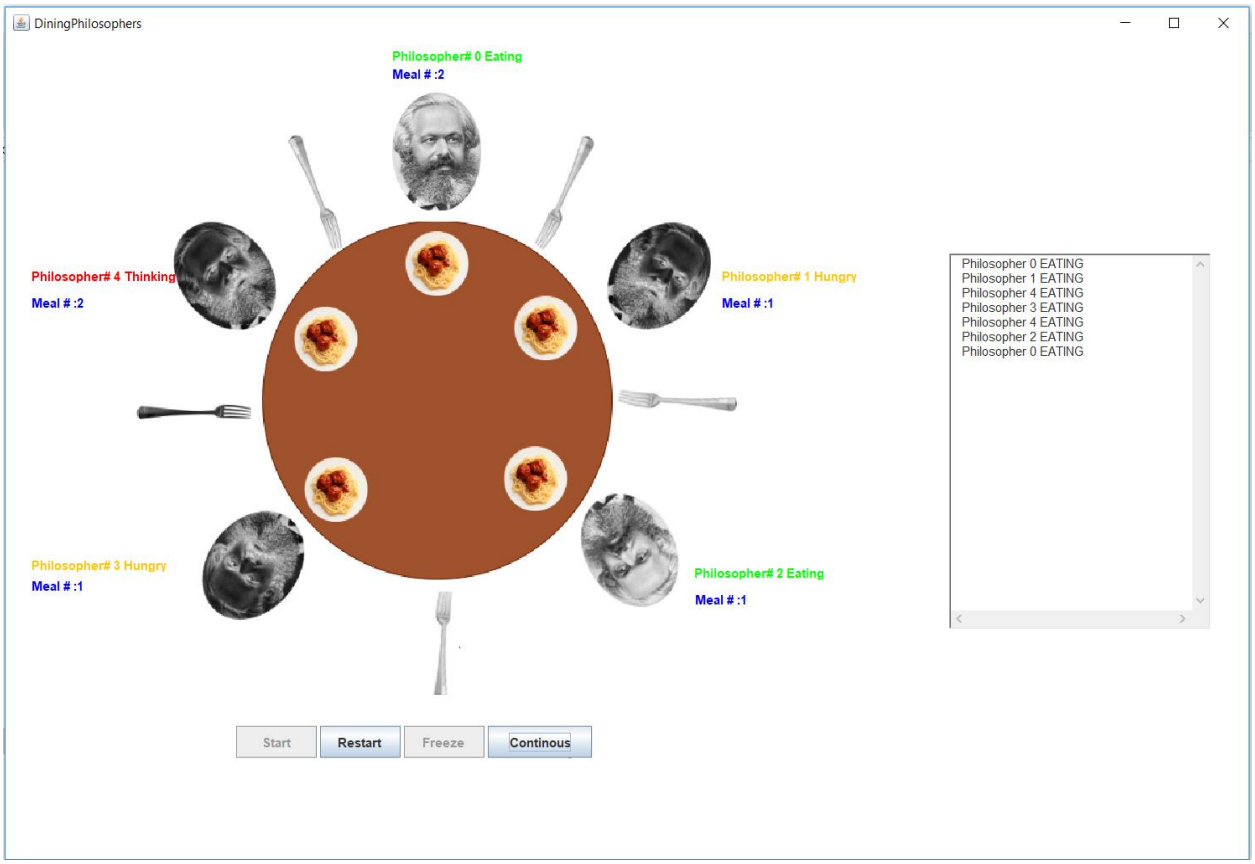
```
private int identity; //Đánh số thức tự của các nĩa
private boolean taken = false; // trạng thái của nĩa
private DiningPhilosophers display; //Cập nhật trạng thái
```

Lớp này có hai phương thức là put() và get(). Phương thức put() được gọi để giải phóng tài nguyên. Phương thức get() được gọi khi một tiến trình muốn sử dụng tài nguyên.

3.2.3. Lớp DiningPhilosophers

- Khởi tạo giao diện
- Khởi tạo các tiến trình
- Tạo môi trường cho các tiến trình hoạt động.

3.3. Kết quả thực thi chương trình



Hình 4. Kết quả thực hiện chương trình bài toán năm triết gia ăn tối.

Tại thời điểm hiện tại như trong hình 4, ta thấy:

- Triết gia số 0 và số 2 đang ăn (Eating).
- Triết gia số 1 và số 3 đang đói bụng (Hungry).
- Triết gia số 4 đang suy nghĩ (Thinking).

Triết gia số 1 đang đói bụng, do là vị trí lẻ nên sẽ ưu tiên lấy chiếc nĩa bên trái trước nhưng do triết gia số 0 đang ăn nên triết gia số 1 phải chờ để số 0 ăn xong đặt đĩa xuống để lấy chiếc đĩa. Triết gia số 3 cũng là vị trí lẻ nên sẽ ưu tiên lấy chiếc bên trái rồi sẽ lấy chiếc bên phải. Nhưng do triết gia số 3 đang ăn nên phải chờ để lấy chiếc nĩa. Để tránh deadlock thì triết gia có thứ tự chẵn sẽ ưu tiên lấy chiếc nĩa bên phải mình trước và ngược lại.

3.4. Đánh giá kết quả

Chương trình đã giải quyết được yêu cầu của bài toán đặt ra. Kết quả thực thi chương trình thỏa mãn tiêu chí như yêu của đề bài.

CHƯƠNG 4. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

4.1. Những điểm đạt được

- Đã giải quyết được cơ bản của vấn đề đặt ra, bao gồm việc giải quyết bài toán Năm triết gia ăn tối và tìm hiểu những khái niệm liên quan đến bài toán đó.
- Tạo ra được các tiến trình mô phỏng dựa trên lớp Thread.
- Chương trình được chia các class với các chức năng rõ ràng tạo thuận lợi cho việc mở rộng và phát triển sau này, đặc biệt phát triển giao diện một cách dễ dàng mà không ảnh hưởng đến phần lõi của chương trình.
- Chương trình hoạt động ổn định.

4.2. Những điểm hạn chế

- Giao diện còn đơn giản, chưa được bắt mắt.

4.3. Hướng phát triển

- Cải tiến giao diện cho bắt mắt và dễ sử dụng hơn.
- Thêm một slide bar để người dùng dễ dàng chỉnh sửa thời gian trung bình của việc ăn và suy nghĩ của triết gia.
- Bắt lỗi chặt hơn trong các chức năng.

PHẦN II: LẬP TRÌNH MẠNG

TIÊU ĐỀ: Sử dụng Socket trong Java xây dựng ứng dụng quản lý danh bạ điện thoại theo mô hình Client_Server.

CHƯƠNG 1: CƠ SỞ LÝ THUYẾT

1.1. Tổng quan về socket

1.1.1.Khái niệm về socket

Một socket là một đầu cuối của một sự truyền thông 2 chiều, liên kết giữa hai chương trình chạy trên mạng. Một socket được gán với một số hiệu cổng(port), vì thế tầng giao vận có thể nhận biết ứng dụng mà dữ liệu được chuyển đến.

Ưu điểm lớn nhất của socket là hỗ trợ hầu hết các hệ điều hành bao gồm MS Windows, Linux,… Ngoài ra, socket cũng được sử dụng với nhiều ngôn ngữ lập trình, gồm C, C++, Java, Visual Basic, Visual C++,… nên nó có thể tương thích với hầu hết mọi đối tượng người dùng với những cấu hình máy khác nhau. Đặc biệt, người dùng cũng có thể chạy cùng một lúc nhiều socket liên tục, giúp nâng cao hiệu suất làm việc, cũng như tiết kiệm thêm nhiều thời gian và công sức hơn.

1.1.2.Phân loại socket

Socket được chia làm 3 loại khác nhau:

- Stream Socket: Dựa trên giao thức TCP(Tranmission Control Protocol), việc truyền dữ liệu chỉ thực hiện giữa 2 quá trình đã thiết lập kết nối. Do đó, hình thức này được gọi là socket hướng kết nối.
- Datagram Socket: Dựa trên giao thức UDP(User Datagram Protocol) việc truyền dữ liệu không yêu cầu có sự thiết lập kết nối giữa 2 quá trình. Do đó, hình thức này được gọi là socket không hướng kết nối.
- Raw Socket: Là socket cho phép truyền và nhận các packet trực tiếp giữa các ứng dụng, bỏ qua tất cả các tầng trung gian.

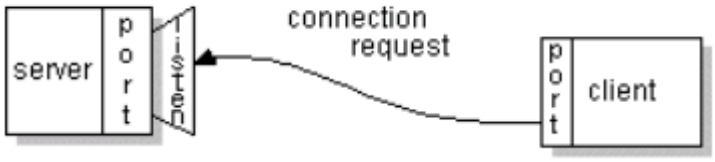
1.1.3.Cơ chế socket trong java

Một socket là một điểm cuối của thông tin hai chiều liên kết giữa hai chương trình đang chạy trên mạng. Những lớp socket được dùng để đại diện cho kết nối giữa một chương trình client và một chương trình server. Trong Java gói Java.net cung cấp hai lớp Socket và ServerSocket để thực hiện kết nối giữa client và server.

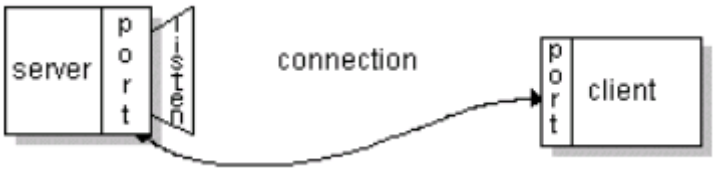
Thông thường thì server sẽ chạy trên một máy đặc biệt và có một socket giới hạn trong 1 Portnumber đặc biệt.

Phía client: client được biết hostname của máy mà server đang chạy và portnumber mà server đang lắng nghe. Để tạo một yêu cầu kết nối client sẽ thử hẹn

gặp server ở trên máy của server thông qua port number. Client cũng cần xác định chính nó với server thông qua local port number.



Hình 5. Client gửi yêu cầu kết nối tới Server



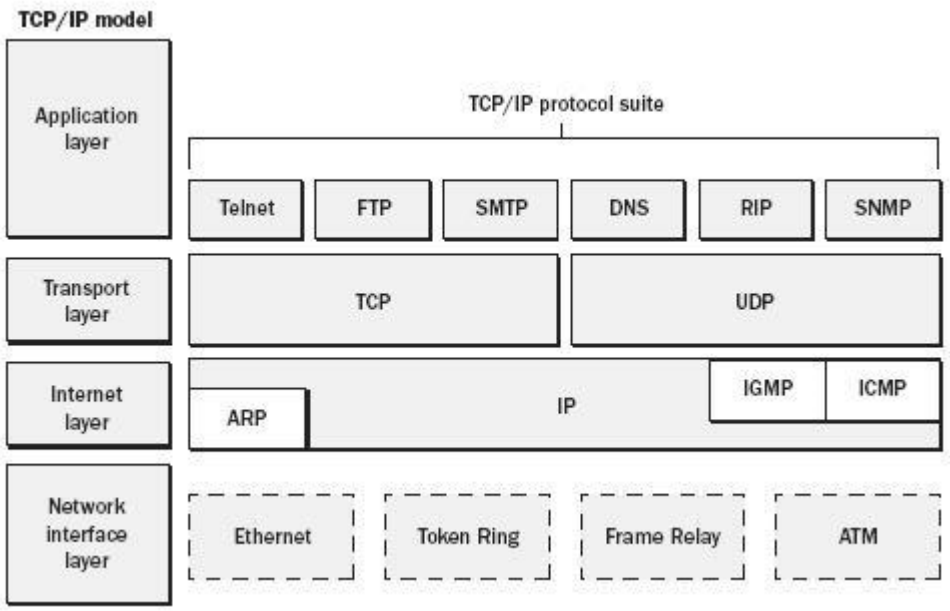
Hình 6. Server đồng ý kết nối và tiếp tục lắng nghe

Nếu mọi thứ tốt đẹp thì server sẽ đồng ý kết nối. khi đồng ý kết nối thì server sẽ tạo ra một socket mới để nói chuyện với client và cũng tạo ra một socket khác để tiếp tục lắng nghe.

1.2. Giao thức TCP/IP

1.2.1.Khái niệm

TCP/IP là viết tắt của cụm từ Transmission Control Protocol/Internet Protocol. TCP/IP là một tập hợp các giao thức (protocol) điều khiển truyền thông giữa tất cả các máy tính trên Internet. Cụ thể hơn, TCP/IP chỉ rõ cách thức đóng gói thông tin (hay còn gọi là gói tin), được gửi và nhận bởi các máy tính có kết nối với nhau. TCP/IP được phát triển vào năm 1978 bởi Bob Kahn và Vint Cerf.



Hình 7. Sơ đồ TCP/IP

1.2.2.Cách thức hoạt động

TCP/IP là sự kết hợp của hai giao thức riêng biệt: Giao thức kiểm soát truyền tin (TCP) và giao thức Internet (IP). Giao thức Internet cho phép các gói được gửi qua mạng; Nó cho biết các gói tin được gửi đi đâu và làm thế nào để đến đó. IP có một phương thức cho phép bất kỳ máy tính nào trên Internet chuyển tiếp gói tin tới một máy tính khác thông qua một hoặc nhiều khoảng (chuyển tiếp) gần với người nhận gói tin.

Giao thức điều khiển truyền dẫn có trách nhiệm đảm bảo việc truyền dữ liệu đáng tin cậy qua các mạng kết nối Internet. TCP kiểm tra các gói dữ liệu xem có lỗi không và gửi yêu cầu truyền lại nếu có lỗi được tìm thấy.

1.3. Mô hình Client/Server

Mô hình được phổ biến nhất và được chấp nhận rộng rãi trong các hệ thống phân tán là mô hình client/server. Trong mô hình này sẽ có một tập các tiến trình mà mỗi tiến trình đóng vai trò như là một trình quản lý tài nguyên cho một tập hợp các tài nguyên cho trước và một tập hợp các tiến trình client trong đó mỗi tiến trình thực hiện một tác vụ nào đó cần truy xuất tới tài nguyên phần cứng hoặc phần mềm dùng chung. Bản thân các trình quản lý tài nguyên cần phải truy xuất tới các tài nguyên dùng chung được quản lý bởi một tiến trình khác, vì vậy một số tiến trình vừa là tiến trình client vừa là tiến trình server. Các tiến trình phát ra các yêu cầu tới các server bất kỳ khi nào chúng cần truy xuất tới một trong các tài nguyên của các server. Nếu yêu cầu là đúng đắn thì server sẽ thực hiện hành động được yêu cầu và gửi một đáp ứng trả lời tới tiến trình client.

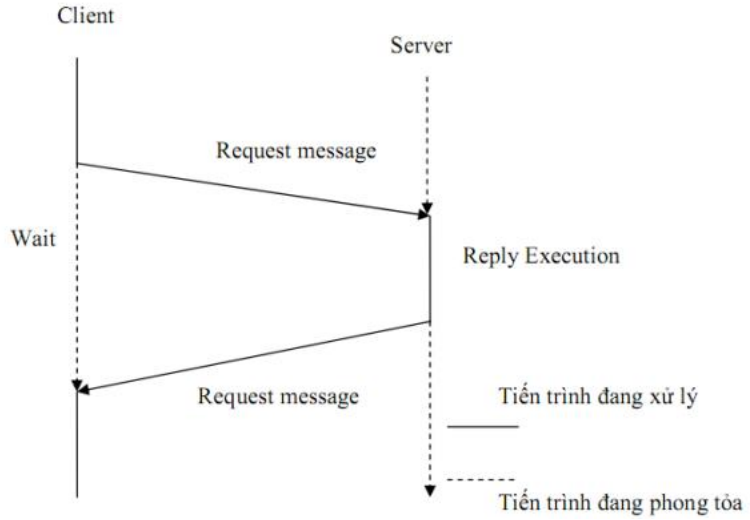
Mô hình client/server cung cấp một cách tiếp cận tổng quát để chia sẻ tài nguyên trong các hệ thống phân tán. Mô hình này có thể được cài đặt bằng rất nhiều môi trường phần cứng và phần mềm khác nhau. Các máy tính được sử dụng để chạy các tiến trình client/server có nhiều kiểu khác nhau và không cần thiết phải phân biệt giữa chúng; cả tiến trình client và tiến trình server đều có thể chạy trên cùng một máy tính. Một tiến trình server có thể sử dụng dịch vụ của một server khác.

Mô hình truyền tin client/server hướng tới việc cung cấp dịch vụ. Quá trình trao đổi dữ liệu bao gồm:

1. Truyền một yêu cầu từ tiến trình client tới tiến trình server
2. Yêu cầu được server xử lý
3. Truyền đáp ứng cho client

Mô hình truyền tin này liên quan đến việc truyền hai thông điệp và một dạng đồng bộ hóa cụ thể giữa client và server. Tiến trình server phải nhận thức được thông điệp được yêu cầu ở bước một ngay khi nó đến và hành động phát ra yêu cầu trong client phải được tạm dừng (bị phong tỏa) và buộc tiến trình client ở trạng thái chờ cho tớ khi nó nhận được đáp ứng do server gửi về ở bước ba.

Mô hình client/server thường được cài đặt dựa trên các thao tác cơ bản là gửi (send) và nhận (receive).



Hình 8. Mô hình Client-Server

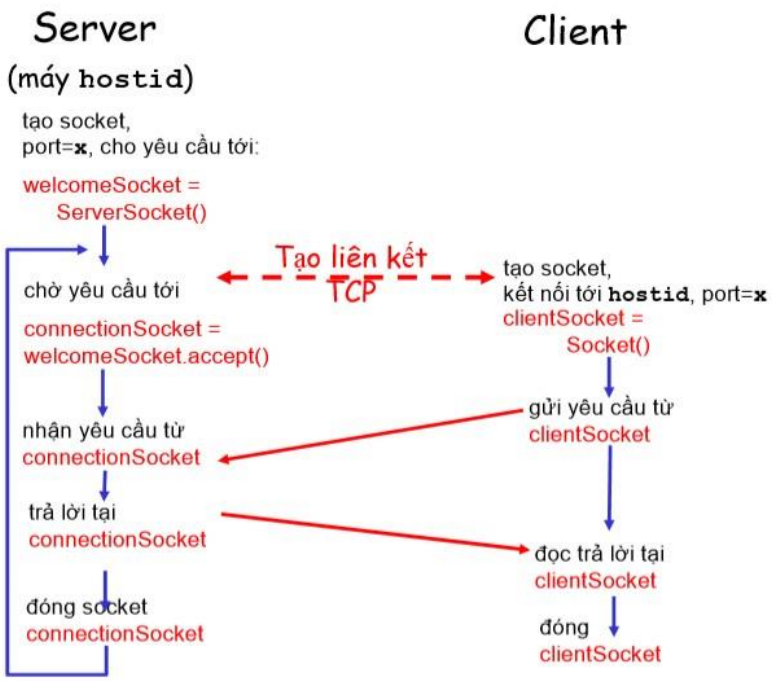
Quá trình giao tiếp client và server có thể diễn ra theo một trong hai chế độ: bị phong tỏa (blocked) và không bị phong tỏa (non-blocked).

- Chế độ bị phong tỏa (blocked): Trong chế độ bị phong tỏa, khi tiến trình client hoặc server phát ra lệnh gửi dữ liệu (send), việc thực thi của tiến trình sẽ bị tạm ngừng cho tới khi tiến trình nhận phát ra lệnh nhận dữ liệu (receive).Tương tự đối với tiến trình nhận dữ liệu, nếu tiến trình nào đó (client hoặc server) phát ra lệnh nhận dữ liệu, mà tại thời điểm đó chưa có dữ liệu gửi tới thì việc thực thi của tiến trình cũng sẽ bị tạm ngừng cho tới khi có dữ liệu gửi tới.
- Chế độ không bị phong tỏa (non-blocked):Trong chế độ này, khi tiến trình client hay server phát ra lệnh gửi dữ liệu thực sự, việc thực thi của tiến trình vẫn được tiến hành mà không quan tâm đến việc có tiến trình nào phát ra lệnh nhận dữ liệu đó hay không.Tương tự cho trường hợp nhận dữ liệu, khi tiến trình phát ra lệnh nhận dữ liệu, nó sẽ nhận dữ liệu hiện có, việc thực thi của tiến trình vẫn được tiến hành mà không quan tâm đến việc có tiến trình nào phát ra lệnh gửi dữ liệu tiếp theo hay không.

1.4. Lập trình TCP Socket với Java

1.4.1. Cách thức hoạt động

Ban đầu, phía server tạo Socket được ràng buộc với một cổng (port number) để chờ nhận yêu cầu từ phía client. Tiếp đến phía client yêu cầu server bằng cách tạo một Socket TCP trên máy kèm với địa chỉ IP và port number của tiến tình tương ứng trên máy server. Khi client tạo Socket, client TCP tạo liên kết với server TCP và chờ chấp nhận kết nối từ server. TCP cung cấp dịch vụ truyền dòng tin cậy và có thứ tự giữa client và server, giữa máy chủ và máy nhận chỉ có 1 địa chỉ IP duy nhất. Thêm vào đó, mỗi thông điệp truyền đi đều có xác nhận trả về.



Hình 9.Mô hình tương tác giữa client và server qua giao thức TCP

1.4.2.Các thư viện trong java lập trình socket chế độ có kết nối (TCP)

1.4.2.1. Lớp java.net.Socket

Lớp java.net.Socket: Hỗ trợ xây dựng chương trình client.

- Socket(String HostName, int PortNumber) throws IOException: nối kết đến Server có tên là HostName, cổng là PortNumber.
  - VD: Socket s = new Socket("www.cit.ctu.edu.vn", 80);

    Hoặc Socket s = new Socket("203.162.36.149", 80);
- InputStream getInputStream() throws IOException: trả về 1 InputStream nối với Socket.
- OutputStream getOutputStream() throws IOException: trả về OutputStream nối với Socket.
  - VD: InputStream is = s.getInputStream();

    OutputStream os = s.getOutputStream();
- void close() throws IOException: đóng Socket lại, giải phóng kênh ảo, xóa nối kết giữa Client và Server.
  - VD: s.close();
- InetAddress getInetAddress(): lấy địa chỉ của máy tính đang nối kết (ở xa).
- int getPort(): lấy cổng của máy tính đang nối kết (ở xa).

1.4.2.2. Lớp java.net.ServerSocket

Lớp java.net.ServerSocket: Hỗ trợ xây dựng chương trình Server.

- ServerSocket(int PortNumber): tạo một Socket của Server và lắng nghe trên cổng PortNumber.
    - VD: ServerSocket ss = new ServerSocket(7);
- Socket accept():Bị nghẽn cho đến khi có một yêu cầu nối kết từ Client. Chấp nhận cho nối kết, trả về một Socket là một đầu của kênh giao tiếp ảo giữa Server và Client.
    - VD: Socket s = ss.accept();
- Server sau đó sẽ lấy InputStream và OutputStream của Socket mới s để giao tiếp với Client:
    - InputStream is = s.getInputStream();

      OutputStream os = s.getOutputStream();.

CHƯƠNG 2: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

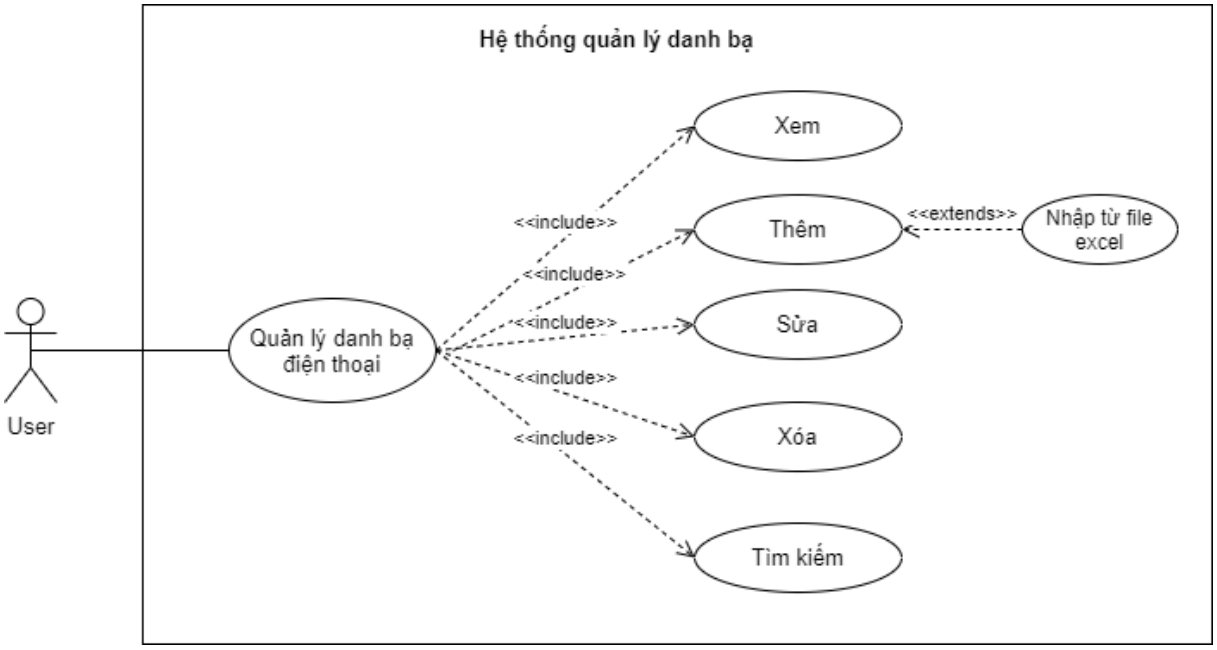
2.1. Phân tích yêu cầu

Sử dụng Socket trong Java xây dựng ứng dụng quản lý danh bạ điện thoại theo mô hình Client_Server.

2.2. Phân tích chức năng

- Xem danh bạ
- Thêm người mới vào danh bạ
- Sửa danh bạ
- Xóa một người trong danh bạ
- Tìm kiếm theo tên, số điện thoại, địa chỉ, email, nhóm

2.3. Sơ đồ usecase tổng quát



Hình 10. Sơ đồ usecase

2.4. Thiết kế hệ thống

2.4.1. Giao thức truyền dữ liệu

Sử dụng giao thức TCP để truyền dữ liệu vì có các ưu điểm:

- Dùng cho mạng WAN.
- Không cho phép mất gói tin.
- Đảm bảo việc truyền dữ liệu.

2.4.2. Lập trình Socket

Xây dựng các chương trình Server sử dụng socket ở chế độ có nối kết. Khởi tạo một ServerSocket với cổng kết nối mặc định là 5500. Phương thức accept() lắng nghe yêu cầu nối kết của các Client. Nó sẽ chờ cho đến khi có một yêu cầu nối kết của client gởi đến. Khi có yêu cầu nối kết của Client gởi đến, nó sẽ chấp nhận yêu cầu nối kết, trả về một Socket là một đầu của kênh giao tiếp ảo giữa Server và Client yêu cầu nối kết.

```
public class Server {
    final static int DEFAULT_PORT = 5500;
    Socket socket;

    public Server() {
        try {
            ServerSocket server = new ServerSocket(DEFAULT_PORT);
            System.out.println("Listening...");
            socket = server.accept();
            System.out.println("Có 1 kết nối đến");
            System.out.println("Connected to " + socket.getInetAddress().getHostAddress());
            new Process(this, socket).start();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public static void main(String[] args) {
        new Server();
    }
}
```

Hình 11. Lập trình socket ở phía server

Xây dựng các chương trình client sử dụng socket ở chế độ có nối kết với HostName của server có giá trị ban đầu mặc định là "localhost", cổng có giá trị mặc định là 5500. Khi kết nối có Server ở máy khác thì HostName sẽ là địa chỉ IP của máy server. Nếu nối kết thành công, một kênh ảo sẽ được hình thành giữa Client và Server.

```
public MainContacts() {
    // TODO Auto-generated constructor stub
    GUI();
    try {
        socket = new Socket(DEFAULT_IP, DEFAULT_PORT);
    } catch (IOException e) {
        // TODO Auto-generated catch block
        System.out.println("Kết nối lỗi");
        lbStatus.setText("Lỗi kết nối server");
        e.printStackTrace();
    }
```

Hình 12. Lập trình socket ở phía client

2.4.3. Cấu trúc chương trình

- Lớp MainContacts: Ở phía client bao gồm các phần giao diện, các hàm thực hiện các chức năng của đề bài, kết nối đến server, gởi yêu cầu và nhận dữ liệu từ server.
- Lớp Server: Bao gồm việc tiếp nhận kết nối từ phía client, kết nối và thực hiện lệnh truy vấn dữ liệu. Sau khi truy vấn gửi dữ liệu về phía client.
- Lớp Person: Gồm các phương thức và thuộc tính đặc trưng cho của lớp person gồm id, name, phone, address, email và group.
- Lớp Data: Cập nhật danh sách danh bạ và danh sách nhóm khi có sự thay đổi trong database.
- Lớp DetailPerson: Xem chi tiết một người trong danh bạ
- Lớp AddPerson: Thực hiện thêm liên hệ.
- Lớp EditPerson: Thực hiện chỉnh sửa thông tin của một liên hệ trong danh bạ.
- Lớp GroupContacts: Gồm các phương thức và thuộc tính đặc trưng cho lớp GroupContacts gồm idGroup và nameGroup.
- Lớp EditGroup: Thực hiện thêm, chỉnh sửa, xóa một nhóm liên lạc trong danh bạ.
- Lớp ReadAndWriteExcel: Thực hiện nhập file dạng excel vào dữ liệu hoặc xuất danh sách danh bạ ra dạng excel theo yêu cầu.
- Lớp sortCodeVN: Sắp xếp danh sách theo thứ tự bảng chữ cái tiếng Việt.

CHƯƠNG 3: TRIỂN KHAI VÀ ĐÁNH GIÁ KẾT QUẢ

3.1. Ngôn ngữ và môi trường cài đặt

3.1.1. Ngôn ngữ

Chương trình được viết bằng ngôn ngữ Java.

3.1.2. Môi trường triển khai

Chạy trên phần mềm Eclipse ở môi trường Windows.

3.2. Triển khai

3.2.1. Class MainContacts

Bao gồm phần giao diện: 1 table để chứa danh sách cácđối tượng, các JtextField, JLabel của tên, số điện thoại, địa chỉ, email, nhóm liên hệ. Các button thêm, sửa, xóa, cập nhật, thoát để gửi các yêu cầu lên Server.Để Client kết nối được với Server cần thông qua socket bao gồm địa chỉ và cổng kết nối. Dữ liệu được gửi và nhận giữa Client và Server thông qua ObjectOutputStream và ObjectInputStream. Sau khi Client nhận dữ liệu Server, dữ liệu được đưa vào bảng để hiển thị lên cửa sổ.

Các phương thức chính trong lớp:

- public void GUI(): Khởi tạo giao diện người dùng tại client.
- public Data getData(): Trả về danh sách liên hệ và danh sách nhóm liên hệ.
- public void setData(Data data): Thiết lập danh sách liên hệ và danh sách nhóm liên hệ.
- public void updateTable(ArrayList<Person> list): Thiết lập bảng hiện thị
- public void updateData() : Cập nhật các thông tin liên lạc
- public void read() : Đọc dữ liệu danh bạ gửi từ server và hiện thị ra bảng.
- private int findIndexSelectContact(): Tìm vị trí hàng được chọn.
- private void editGroupContacts() : Chỉnh sửa nhóm liên hệ
- private void addContact(): Thêm liên hệ
- public void editContact(): Chỉnh sửa thông tin liên hệ được chọn
- private void detailPerson(): Hiện thị thông tin chi tiết một liên hệ được chọn.
- private void delete() : Xóa một liên hệ được chọn.
- private ArrayList<Person> search(int typeSearch): Trả về danh sách liên hệ tìm kiếm.
- private void searchStatus(int count, String status, String textFind): Thiết lập trạng thái tìm kiếm.

- private void resetSearch(): Thiết lập lại thông tin tìm kiếm.
- private void sort() : Sắp xếp bảng theo cột được chọn.

3.2.2. Class Server

Lặp liên tục để mở cổng kết nối và chấp nhận kết nối từ nhiều Client đến Server. Sử dụng JDBC kết nối với cơ sở dữ liệu. Để kết nối được với cơ sở dự liệu phía Server cần có:

- Drivermanager: là class quản lý danh sách các driver.
- Driver: để liên lạc với database.
- Connection: thông tin của database bao gồm: url, user, password.
- Statement: là các tập lệnh SQL để truy vấn dữ liệu.
- Resultset: là tập các bản ghi nhận được sau khi thực hiện truy vấn.

3.2.3. Class Person

Các phương thức chính trong lớp:

- public int getId(): Trả về id của đối tượng.
- public void setId(int id): Thiết lập id của đối tượng.
- public String getName(): Trả về tên của đối tượng.
- public void setName(String name): Thiết lập tên của đối tượng.
- public String getAddress(): Trả về địa chỉ của đối tượng.
- public void setAddress(String address): Thiết lập địa chỉ của đối tượng.
- public String getPhone(): Trả về số điện thoại của đối tượng.
- public void setPhone(String phone): Thiết lập số điện thoại của đối tượng
- public String getEmail(): Trả về email của đối tượng.
- public void setEmail(String email): Thiết lập email của đối tượng.
- public String getGroup(): Trả về tên của nhóm liên lạc.
- public void setGroup(String group): Thiết lập tên của nhóm liên lạc
- public GroupContact getClassify(): Trả về nhóm của đối tượng.
- public void setClassify(GroupContact classify): Thiết lập nhóm của đối tượng.
- public int getIdClassify(): Lấy id nhóm của đối tượng.

3.2.4. Class Data

Các phương thức chính trong lớp:

- public ArrayList<Person> getlistPerson(): Trả về danh sách liên hệ.

- public void setlistPerson(ArrayList<Person> listPerson): Thiết lập danh sách liên hệ.
- public ArrayList<String> getListGroup(): Trả về danh sách nhóm liên hệ.
- public void setListGroup(ArrayList<String> listGroup): Thiết lập danh sách nhóm liên hệ.
- public ArrayList<GroupContact> getListGroupNew(): Trả về danh sách nhóm liên hệ mới.
- public void setListGroupNew(ArrayList<GroupContact> listGroupNew): Thiết lập danh sách nhóm liên hệ mới.
- public ArrayList<String> getListGroupName():Trả về danh sách tên của các nhóm liên hệ.

3.2.5. Class DetailPerson

Hiện thị thông tin chi tiết của một liên lạc. Các phương thức chính trong lớp:

- private void GUI(): Giao diện thông tin chi tiết liên hệ
- private void loadInfor(): Hiện thị các thông tin ra giao diện
- public int getIndexSelected(): Trả về vị trí được chọn
- public void setIndexSelected(int indexSelected): Thiết lập vị trí được chọn trong bảng.
- public void display(boolean visible): Thiết lập hiện thị giao diện

3.2.6. Class AddPerson

Các phương thức chính trong lớp:

- public void GUI(): Giao diện thêm một liên lạc.
- public Person getInfor(): Trả về các thông tin của đối tượng được nhập vào.
- private boolean checkTextField(JTextField txt): Kiểm tra tính hợp lệ của các ô JtextField.
- private boolean checkComboBox(JComboBox<String> cb): Kiểm tra tính hợp lệ của JcomboBox.
- public int indexGroupName(String groupName): Trả về vị trí của tên nhóm được chọn.
- private void add(): Gửi thông tin thêm vào danh bạ lên server.
- private void retypeData(): Yêu cầu nhập lại dữ liệu.
- private void cancel(): Hủy thêm liên hệ.
- private void clearInput() : Xóa những gì đã nhập.

- public void setListGroupNew(ArrayList<GroupContact> listGroupNew): Thiết lập danh sách nhóm mới được thêm vào.
- private void loadInfor(): Thiết lập danh sách nhóm mới về giao diện chính.
- public void loadListGroup(): Hiện danh sách nhóm liên hệ
- public void loadListGroupNew(): Hiện thị danh sách nhóm liên hệ mới
- private void addGroup(String groupName): Gửi lên server yêu cầu thêm một nhóm liên hệ.
- public void display(boolean visible): Thiết lập hiện thị giao diện thêm.

3.2.7. Class EditPerson

Các phương thức chính trong lớp:

- private void GUI(): Giao diện chỉnh sửa thông tin.
- private void loadInfor(): Hiện thị các thông tin ra giao diện
- public int getIndexSelected(): Trả về vị trí được chọn
- public void setIndexSelected(int indexSelected): Thiết lập vị trí được chọn trong bảng.
- public void display(boolean visible): Thiết lập hiện thị giao diện
- public void loadListGroup(): Hiện danh sách nhóm liên hệ
- public void loadListGroupNew(): Hiện thị danh sách nhóm liên hệ mới
- public Person getInfor(): Trả về các thông tin của đối tượng được nhập vào.
- private void editPerson(): Gửi thông tin chỉnh sửa lên server.
- private void addGroup(String groupName): Gửi lên server yêu cầu thêm một nhóm liên hệ.
- private void cancel(): Hủy chỉnh sửa liên hệ.
- private void clearInput() : Xóa những gì đã nhập.

3.2.8. Class GroupContacts

Các phương thức chính trong lớp:

- public int getId(): Trả về id của nhóm liên lạc
- public void setId(int id): Thiết lập id của nhóm liên lạc.
- public String getName(): Lấy tên của nhóm
- public void setName(String groupName): Thiết lập tên của nhóm

3.2.9. Class EditGroup.

Các phương thức chính trong lớp:

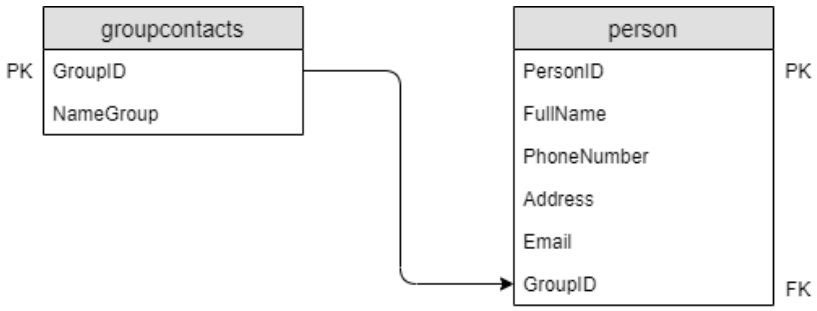
- private void GUI(): Giao diện chỉnh sửa nhóm.
- private void loadInfor(): Hiện thị các thông tin ra giao diện.
- public void display(boolean visible): Thiết lập hiện thị giao diện
- public void loadListGroup(): Hiện danh sách nhóm liên hệ
- public void loadListGroupNew(): Hiện thị danh sách nhóm liên hệ mới
- public void setListGroupNew(ArrayList<GroupContact> listGroupNew): Thiết lập danh sách nhóm mới được thêm vào.
- public int indexGroupName(String groupName): Trả về vị trí của tên nhóm được chọn.
- private void addGroup(String groupName): Gửi lên server yêu cầu thêm một nhóm liên hệ.
- private void delete(): Xóa một nhóm liên lạc.
- private void renameGroup(): Đổi tên nhóm liên lạc.
- public void setName(String groupName): Thiết lập tên của nhóm

3.2.10. Class EditGroup.

- public ArrayList<Person> ReadFile(File file): Đọc file excels.
- public boolean SaveContacts(ArrayList<Person> listPersons): Lưu dữ liệu thành file excel.

3.2.11.Database.

Sử dụng MySQL để tạo database:



Hình 13. Quan hệ giữa các bảng trong database

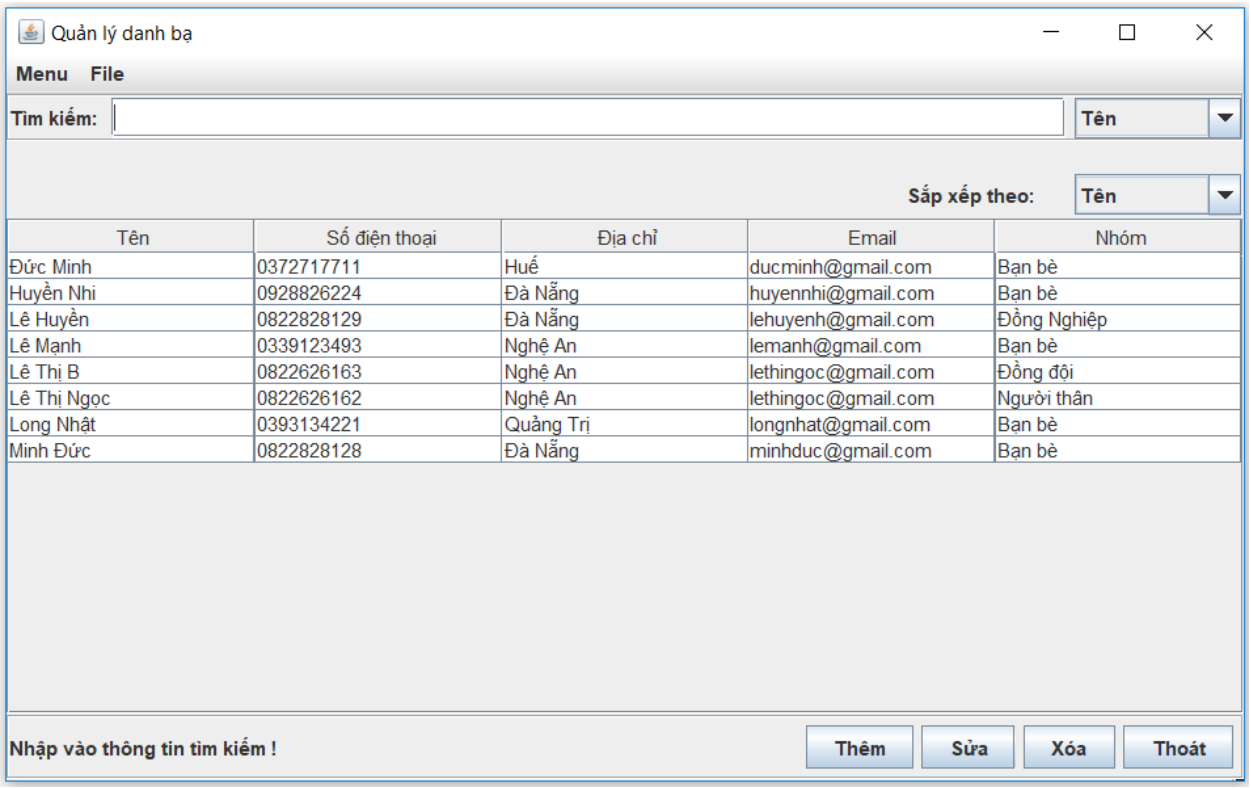
PersonID	FullName	PhoneNumber	Address	Email	GroupID
24	Lê Mạnh	0339123493	Nghệ An	lemanh@gmail.com	38
25	Long Nhật	0393134221	Quảng Trị	longnhat@gmail.com	38
26	Đức Minh	0372717711	Huế	ducminh@gmail.com	38
28	Lê Thị Ngọc	0822626162	Nghệ An	lethingoc@gmail.com	39
29	Minh Đức	0822828128	Đà Nẵng	minhduc@gmail.com	38
30	Lê Huyền	098282221242	Đà Nẵng	lehuyenh@gmail.com	40
31	Huyền Nhi	0928826224	Đà Nẵng	huyennhi@gmail.com	38
NULL	NULL	NULL	NULL	NULL	NULL

Hình 14. Bảng database person

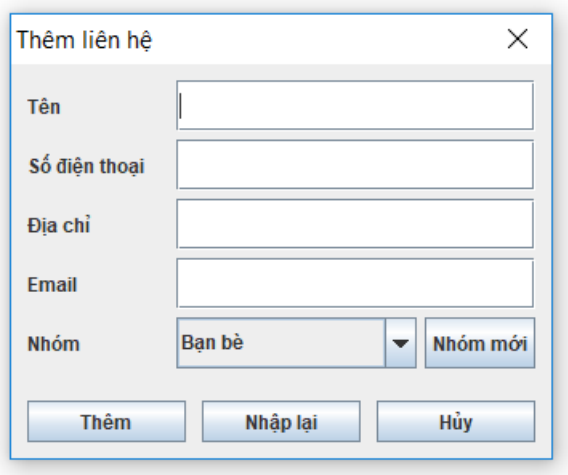
	GroupID	NameGroup
▶	38	Bạn bè
	39	Người thân
	40	Đồng Nghiệp
*	NULL	NULL

Hình 15. Bảng database groupcontacts

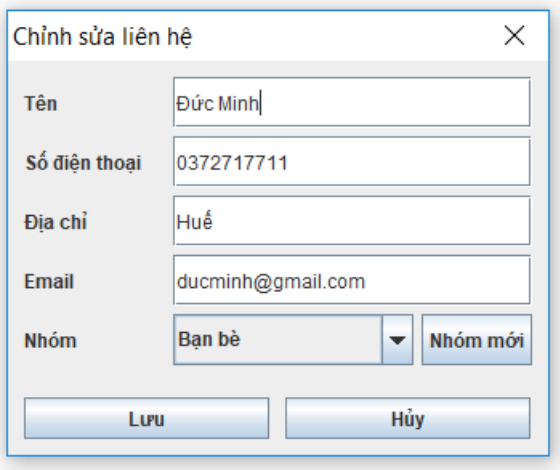
3.3. Kết quả thực thi chương trình



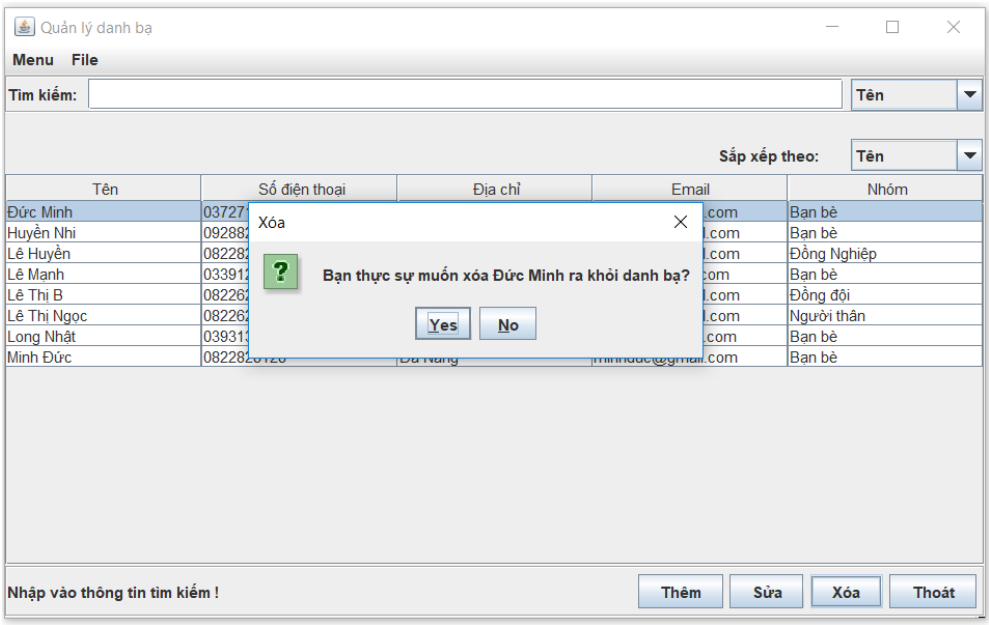
Hình 16. Màn hình giao diện chính của chương trình



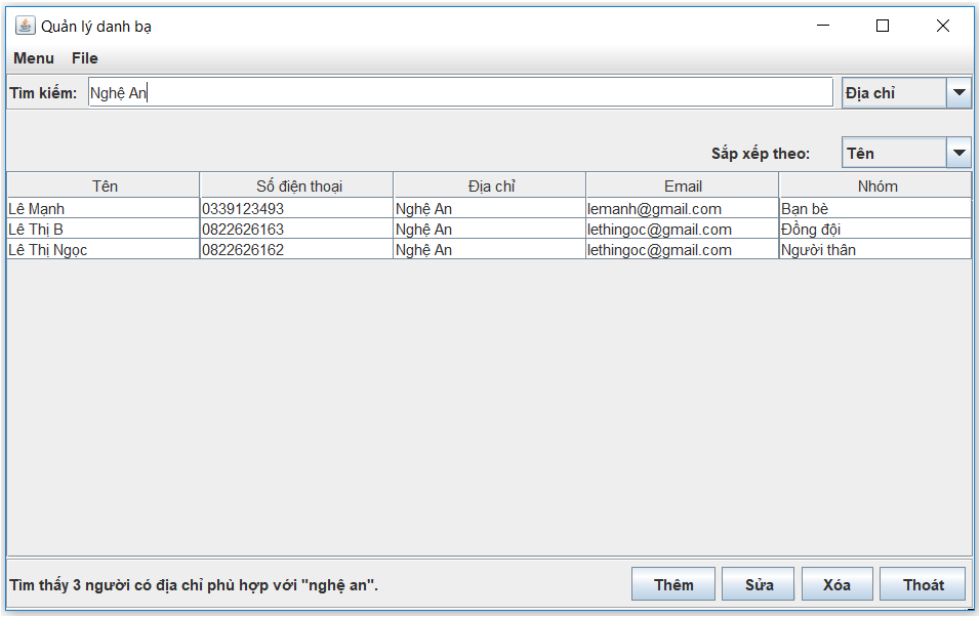
Hình 17. Màn hình giao diện thêm liên hệ



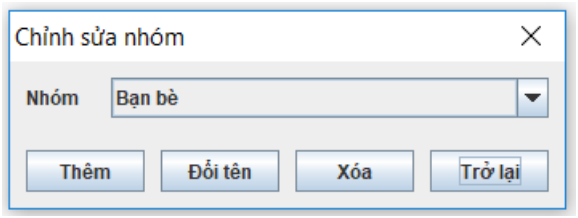
Hình 18. Màn hình giao diện chỉnh sửa liên hệ



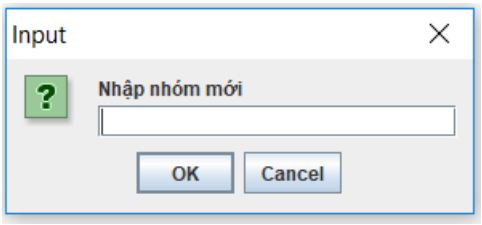
Hình 19. Màn hình giao diện xóa liên hệ



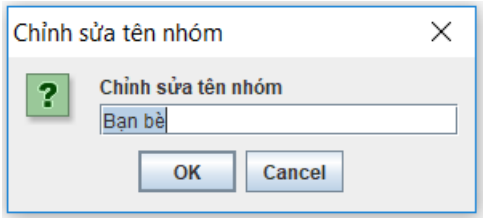
Hình 20. Màn hình giao diện kết quả tìm kiếm theo địa chỉ "Nghệ An"



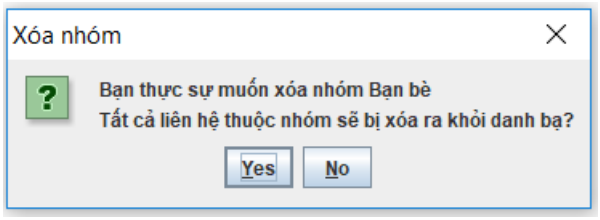
Hình 21. Màn hình giao diện chỉnh sửa nhóm



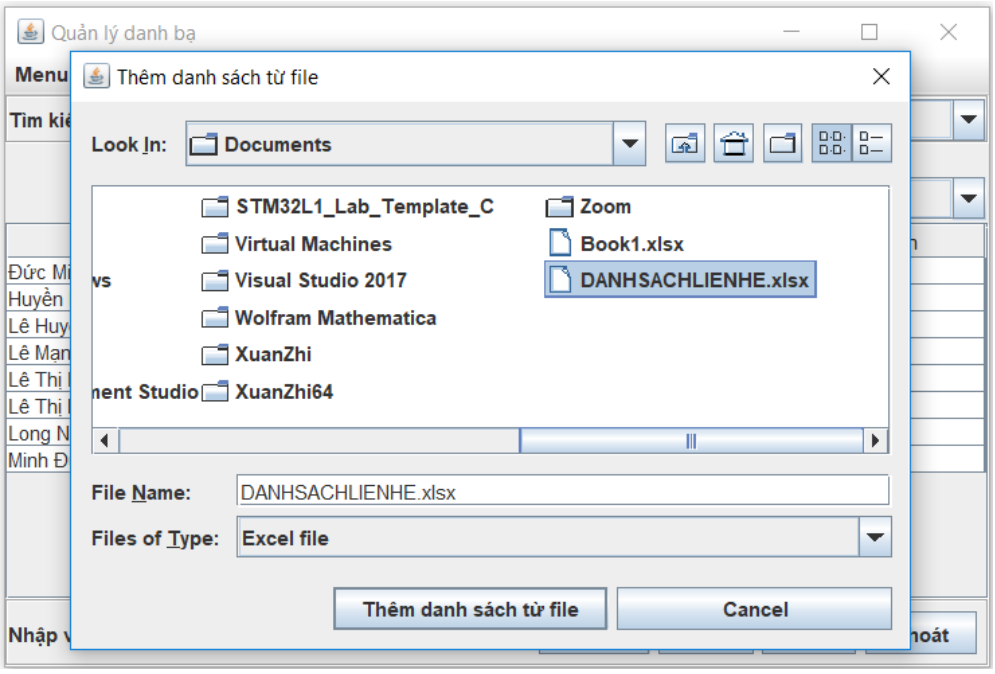
Hình 22. Màn hình giao diện thêm nhóm



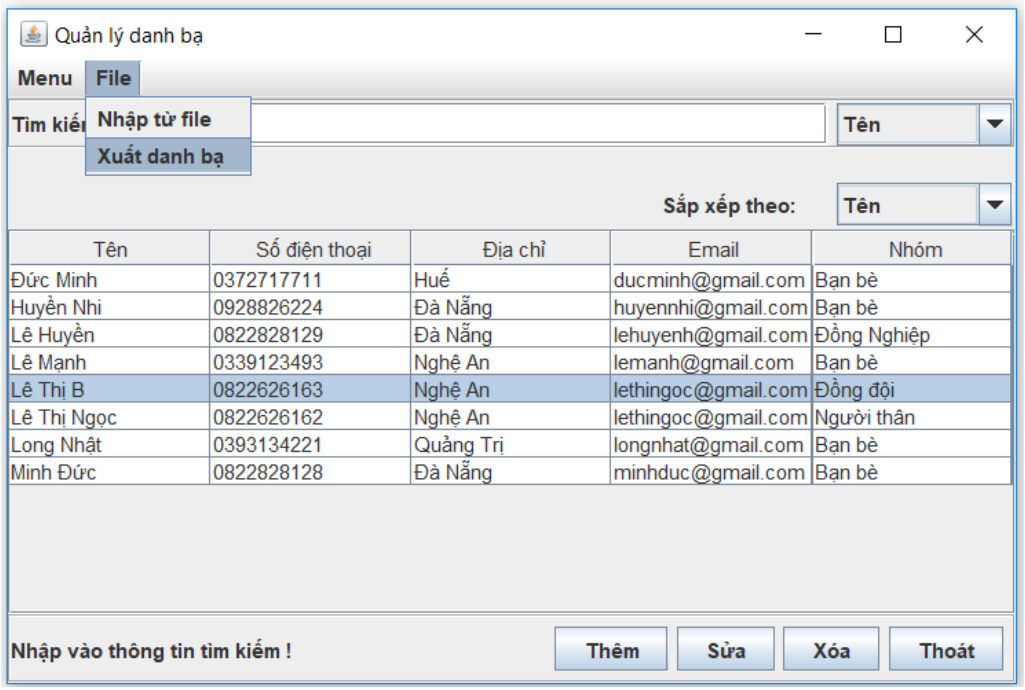
Hình 23. Màn hình giao diện chỉnh sửa tên nhóm



Hình 24. Màn hình giao diện xóa nhóm



Hình 25. Màn hình thêm liên hệ từ file excel



Hình 26. Màn hình xuất danh sách liên hệ ra file excel

3.5. Đánh giá kết quả.

Chương trình được viết bằng ngôn ngữ Java hỗ trợ dễ dàng lập trình giao thức TCP. Có các thư viện hỗ trợ gửi và nhận dữ liệu bằng socket một cách nhanh chóng. Người lập trình có thể đọc hiểu code một cách dễ hiểu. Ngoài ra java có cơ chế bắt ngoại lệ mạnh giúp chương trình hoạt động hiểu quả hơn và kết nối cơ sở dữ liệu một cách đơn giản và nhanh chóng.

CHƯƠNG 4. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

4.4. Những điểm đạt được

- Hiểu được cơ chế hoạt động socket, kết nối cơ sở dữ liệu.
- Hiểu được cách lập trình cơ bản các chức năng quản lý một danh bạ bằng ngôn ngữ lập trình java
- Chương trình hoạt động ổn định.

4.5. Những điểm hạn chế

- Giao diện còn đơn giản, khó thao tác.
- Còn thiếu xót trong xử lý bắt lỗi các trường ngoại lệ trong một số chức năng của chương trình

4.6. Hướng phát triển

- Cải tiến giao diện cho dễ sử dụng,
- Bắt lỗi chặt hơn trong các chức năng.

KẾT LUẬN CHUNG

Trong quá trình thực hiện đề tài, nhờ sự giúp đỡ tận tình từ Cô Trần Hồ Thủy Tiên, đồ án đã đạt được một số kết quả nhất định theo đúng yêu cầu đề tài.

Qua đồ án cơ sở mạng này đã giúp em nghiên cứu sâu hơn về nguyên lý hệ điều hành, hiểu rõ cách hệ điều hành cấp phát tài nguyên cũng như có thể xác định được trạng thái của hệ thống. Đồ án đã giúp củng cố, bổ sung thêm kiến thức và luyện tập khả năng lập trình ngôn ngữ Java. Đồng thời giúp em hiểu rõ hơn về giao thức TCP và cách xây dựng nên một ứng dụng quản lý theo mô hình Client-Server, kết nối cơ sở dữ liệu, từ đó xây dựng nên các những ứng dụng khác lớn hơn.

Vì thời gian nghiên cứu không nhiều, điều kiện không cho phép và khả năng còn hạn chết nên kết quả thu được có thể có những thiếu sót không mong muốn, kính mong sự góp ý từ các thầy cô để hoàn thiện đồ án tốt hơn. Em xin chân thành cảm ơn thầy cô khoa công nghệ thông tin đã tạo điều kiện cho em có cơ hội để thực hiện đồ án này.

TÀI LIỆU THAM KHẢO

[1] Link tham khảo: https://viblo.asia/p/tim-hieu-ve-dong-bo-luong-trong-java-voi-bai-toan-bua-toi-cua-cac-triet-gia-OeVKBdMrlkW

[2] Link tham khảo: https://vi.wikipedia.org/wiki/Tiến_trình_(khoa_hoc_máy_tính)

[3] Báo cáo đồ án nguyên lý hệ điều hành khóa 06

[4] Trần Hồ Thuỷ Tiên, Bài giảng Nguyên lý hệ điều hành, Khoa CNTT Trường Đại học Bách khoa Đà Nẵng.

[5] Bài giảng Nguyên lý hệ điều hành- Trường Đại Học Hàng Hải.

[6] Bài giảng Nguyên lý hệ điều hành- Trường Đại Học Mở Hà Nội.

[7] Giáo trình Lập trình mạng, thầy Phạm Minh Tuấn, Mai Văn Hà, Khoa CNTT Đại học BKDN.

[8] Link tham khảo: https://viblo.asia/p/networking-socket-hoat-dong-nhu-the-nao-aWj53LxYK6m

[9] Link tham khảo: https://viettuts.vn/lap-trinh-mang-voi-java/giao-thuc-tcp-ip

[10] Link tham khảo: https://mobilesprogramming.wordpress.com/2010/07/18/lập-trinh-mang-trong-javaphần-5/

[11] Slide bài giảng Công Nghệ Web, thầy Mai Văn Hà, Khoa CNTT Đại học BKDN.

[12] Link tham khảo : https://stackoverflow.com.

[13] Link tham khảo: https://cachhoc.net/2014/06/11/java-demo-quan-ly-danh-ba/