LỜI CẢM ƠN

Để hoàn thành đồ án tốt nghiệp này, lời đầu tiên em xin chân thành cảm ơn các thầy giáo, cô giáo Khoa Công nghệ thông tin Trường Đại học Dân lập Hải Phòng, những người đã dạy dỗ, trang bị cho em những kiến thức bổ ích trong bốn năm học vừa qua.

Em xin bày tỏ lòng biết ơn sâu sắc nhất tới thầy giáo Phùng Anh Tuấn, người đã tận tình hướng dẫn, chỉ bảo em trong suốt thời gian thực tập và làm đồ án.

Nhân dịp này em xin gửi lời cảm ơn chân thành tới gia đình, bạn bè, những người thân đã cổ vũ, động viên tiếp thêm cho em nghị lực để em hoàn thành đồ án tốt nghiệp.

Em xin chân thành cảm ơn !

Hải Phòng, ngày 01 tháng 07 năm 2009

Sinh viên

Nguyễn Thị Hoàng

MỤC LỤC

LỜI MỞ ĐẦU

Ngày nay, với sự phát triển với tốc độ chóng mặt của khoa học kỹ thuật, một kỷ nguyên mới được mở ra, kỷ nguyên của công nghệ thông tin. Nhu cầu của loài người ngày càng lớn, đặc biệt là các ngành khoa học kỹ thuật khác đều cần đến sự hỗ trợ của công nghệ thông tin, mặc dù công nghệ phần cứng phát triển rất nhanh, CPU với tốc độ xử lý ngày càng cao, nhưng lại nảy sinh nhiều bài toán trong thực tế sản xuất đòi hỏi phải xử lí nhanh hơn nữa.

Vấn đề xử lý song song đang ngày càng được nghiên cứu nhiều để giải quyết một số bài toán mà thực tiễn đang đặt ra, những vấn đề cần có kết quả trong thời gian thực như: bài toán dự báo thời tiết, điều tiết giao thông, điều khiển các con tàu vũ trụ,các bài toán về mô phỏng…Vì vậy, việc nghiên cứu các giải thuật cho xử lý song song là một yêu cầu và là một thách thức cho các nhà khoa học liên quan đến khoa học máy tính. Java ra đời trong sự dự đoán trước những gì sẽ xảy ra trong thế giới của công nghệ máy tính, nó hỗ trợ cho việc xử lý song song với cơ chế đa luồng.

Nhưng trong lĩnh vực giáo dục thì lượng tài liệu nói về lập trình đa luồng nói chung và lập trình đa luồng trong Java còn tương đối ít và trình bầy chưa sâu, nhất là các ví dụ minh họa cho cơ chế lập trình này có thể nói là hiếm. Nội dung đồ án tốt nghiệp này cố gắng làm rõ một số khái niệm cơ bản của lập trình đa luồng trong Java và cài đặt chương trình ứng dụng minh họa.

Nội dung đồ án tốt nghiệp được trình bầy trong 5 chương

Chương 1 trình bầy những kiến thức căn bản về mạng máy tính: định nghĩa, phân loại, các loại giao thức mạng, các mô hình hoạt động của mạng máy tính,…để ta có thể tiếp cận với các chương tiếp theo.

Chương 2 giới thiệu về Java, các tính chất, các dạng chương trình ứng dụng của Java, cấu trúc của tệp chương trình Java.

Chương 3 trình bầy về lập trình Socket TCP và lập trình Socket TCP trong Java.

Chương 4 giới thiệu khái niệm luồng, các cách tiếp cận luồng, từ đó đi sâu vào các vấn đề liên quan đến luồng trong Java: các phương pháp thực hiện, độ ưu tiên, nhóm luồng, đồng bộ hóa các luồng thi hành.

Chương 5 trình bầy chi tiết ứng dụng truy nhập cơ sở dữ liệu web dựa trên việc tìm hiểu lý thuyết lập trình đa luồng trong Java

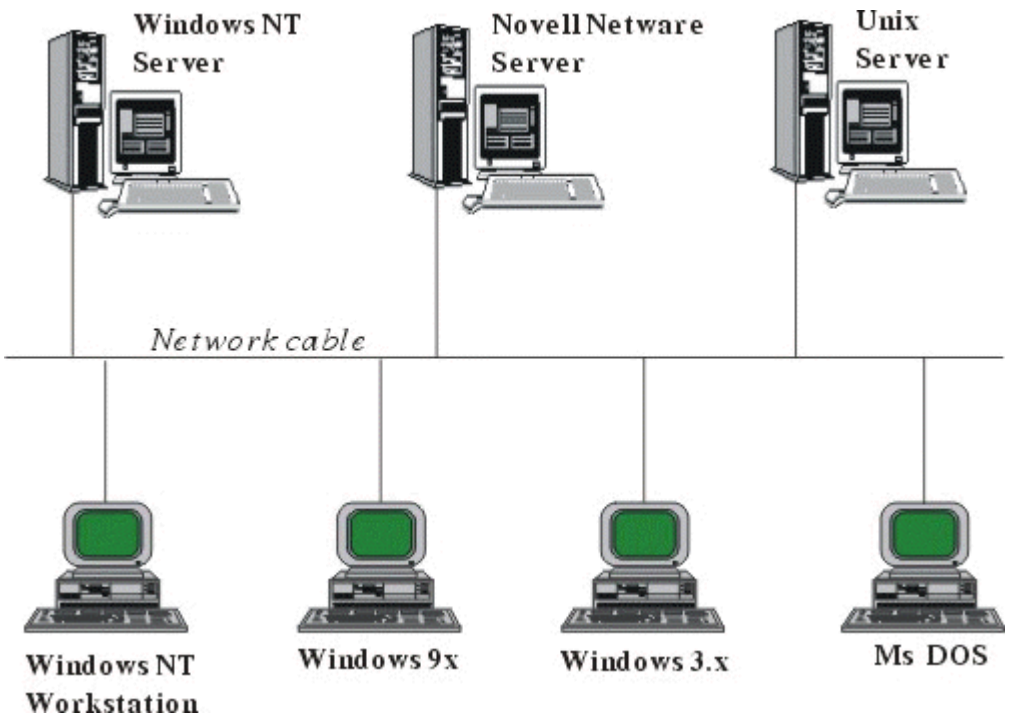
Tiếp theo là phần kết luận, cuối cùng là tài liệu tham khảo và phụ lục.

CHƯƠNG 1: CĂN BẢN VỀ MẠNG MÁY TÍNH

1.1. Định nghĩa mạng máy tính

Mạng máy tính là một tập hợp các máy tính được nối với nhau bởi đường truyền theo một cấu trúc nào đó và thông qua đó các máy tính trao đổi thông tin qua lại cho nhau.

Đường truyền là hệ thống các thiết bị truyền dẫn có dây hay không dây dùng để chuyển các tín hiệu điện tử từ máy tính này đến máy tính khác. Các tín hiệu điện tử đó biểu thị các giá trị dữ liệu dưới dạng các xung nhị phân (on - off). Tất cả các tín hiệu được truyền giữa các máy tính đều thuộc một dạng sóng điện từ. Tùy theo tần số của sóng điện từ có thể dùng các đường truyền vật lý khác nhau để truyền các tín hiệu. Ở đây đường truyền được kết nối có thể là dây cáp đồng trục, cáp xoắn, cáp quang, dây điện thoại, sóng vô tuyến, ... Các đường truyền dữ liệu tạo nên cấu trúc của mạng. Hai khái niệm đường truyền và cấu trúc là những đặc trưng cơ bản của mạng máy tính.



Hình 1.1. Một mô hình các máy tính liên kết trong mạng

1.2. Nhu cầu phát triển mạng máy tính

Ngày nay, khi máy tính được sử dụng một cách rộng rãi và số lượng máy tính trong một văn phòng hay cơ quan được tăng lên nhanh chóng thì việc kết nối chúng trở nên vô cùng cần thiết và sẽ mang lại nhiều hiệu quả cho người sử dụng.

Với một lượng lớn về thông tin, nhu cầu xử lý thông tin ngày càng cao, mạng máy tính đã trở nên quá quen thuộc đối với chúng ta trong mọi lĩnh vực như: thương mại, dịch vụ, giáo dục, khoa học, quân sự, quốc phòng, ...

Người ta thấy được việc kết nối các máy tính thành mạng cho chúng ta những khả năng mới to lớn như:

- *Sử dụng chung tài nguyên:* những tài nguyên (như thiết bị, chương trình, dữ liệu) khi được trở thành các tài nguyên chung thì mọi thành viên của mạng đều có thể tiếp cận được mà không cần quan tâm tới những tài nguyên đó ở đâu.

- *Tăng độ tin cậy của hệ thống:* người ta có thể dễ dàng bảo trì máy móc, lưu trữ (backup) các dữ liệu chung và khi có trục trặc trong hệ thống thì chúng có thể được khôi phục nhanh chóng. Trong trường hợp có trục trặc trên một trạm làm việc thì người ta cũng có thể sử dụng những trạm khác thay thế.

- *Nâng cao chất lượng và hiệu quả khai thác thông tin:* khi thông tin có thể được sử dụng chung thì nó mang lại cho người sử dụng khả năng tổ chức lại các công việc với những thay đổi về chất như:

+ Đáp ứng những nhu cầu của hệ thống ứng dụng kinh doanh hiện đại.

+ Cung cấp sự thống nhất giữa các dữ liệu.

+ Tăng cường năng lực xử lý nhờ kết hợp các bộ phận phân tán.

+ Tăng cường truy nhập tới các dịch vụ mạng khác nhau đang được cung cấp trên thế giới.

Với nhu cầu đòi hỏi ngày càng cao của xã hội nên vấn đề kỹ thuật trong mạng là mối quan tâm hàng đầu của các nhà tin học. Ví dụ như: làm thế nào để truy xuất thông tin một cách nhanh chóng và tối ưu, trong khi việc xử lý thông tin trên mạng quá nhiều, đôi khi có thể làm tắc nghẽn và gây ra mất thông tin một cách đáng tiếc. Hiện nay, việc làm sao có được một hệ thống mạng chạy thật tốt, thật an toàn với lợi ích kinh tế cao đang rất được quan tâm.

1.3. Phân loại mạng máy tính

Do hiện nay mạng máy tính được phát triển khắp nơi với những ứng dụng ngày càng đa dạng cho nên việc phân loại mạng máy tính là một việc rất phức tạp.

Dựa theo phạm vi phân bố của mạng ta có thể phân ra các loại mạng như sau:

- **GAN** (Globa Area Network): kết nối máy tính giữa các châu lục với nhau thông qua mạng viễn thông và vệ tinh.

- **WAN** (Wide Area NetWork): kết nối máy tính trong nội bộ các quốc gia hay giữa các quốc gia trong một châu lục; việc thực hiện kết nối thông qua mạng viễn thông.

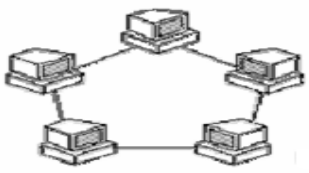
- **MAN** (Metropolitan Area Network): kết nối máy tính trong phạm vi một thành phố. Kết nối này được thực hiện thông qua môi trường truyền thông tốc độ cao (50-100 Mbps).

- **LAN** (Local Area Network): là mạng cục bộ kết nối các máy tính trong khu vực bán kính hẹp (thông thường khoảng vài trăm mét). Kết nối được thực hiện trong môi trường truyền thông tốc độ cao. LAN thường được sử dụng trong một cơ quan hay một tổ chức, do vậy mạng LAN được sử dụng rất phổ biến.

1.4. Một số topo mạng thông dụng

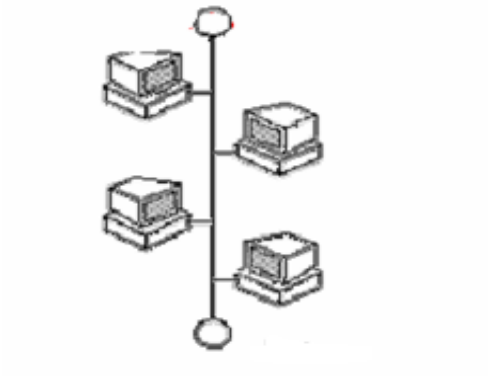
Theo định nghĩa về mạng máy tính, các máy tính được nối với nhau bởi các đường truyền vật lý theo một kiến trúc nào đó, các kiến trúc đó gọi là Topology. Thông thường mạng có ba loại kiến trúc, đó là: mạng hình sao (Star Topology), mạng dạng tuyến (Bus Topology), mạng dạng vòng (Ring Topology).

- *Ring Topology*: Mạng được bố trí vòng tròn, đường dây cáp được thiết kế làm thành một vòng khép kín, tín hiệu chạy theo một chiều nào đó. Các nút truyền tín hiệu cho nhau tại một thời điểm được một nút mà thôi. Mạng dạng vòng có thuận lợi là có thể nới rộng ra xa nhưng đường dây phải khép kín, nếu bị ngắt ở một nơi nào đó thì toàn bộ hệ thống cũng bị ngưng.

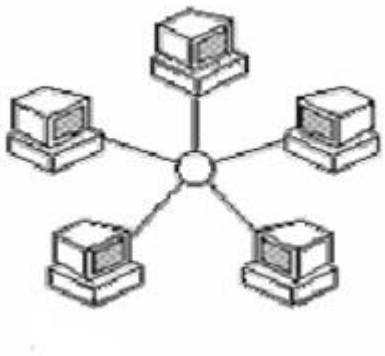


Hình 1.2. Ring Topology

- *Bus Topology*: Ở dạng Bus tất cả các nút được phân chia một đường truyền chính (bus). Đường truyền này được giới hạn hai đầu bởi một loại đầu nối đặc biệt gọi là Terminator. Khi một nút truyền dữ liệu, tín hiệu được quảng bá trên hai chiều của bus, mọi nút còn lại đều được nhận tín hiệu trực tiếp. Loại mạng này dùng dây cáp ít, dễ lắp đặt. Tuy vậy cũng có những bất lợi, đó là sẽ có sự ùn tắc giao thông khi di chuyển với lưu lượng lớn và khi có sự hỏng hóc ở đoạn nào đó thì rất khó phát hiện, nếu một nút ngừng hoạt động sẽ ảnh hưởng tới toàn bộ hệ thống.



Hình 1.3. Bus Topology



Hình 1.4. Star Topology

- *Star Topology*: Mạng hình sao bao gồm một bộ tập trung và các nút thông tin. Các nút thông tin có thể là các trạm cuối, các máy tính hay các thiết bị khác của mạng. Mạng hoạt động theo nguyên lý nối song song nên nếu có một nút bị hỏng, mạng vẫn hoạt động bình thường. Mạng có thể mở rộng hoặc thu hẹp tùy theo yêu cầu của người sử dụng, tuy nhiên mở rộng phụ thuộc vào khả năng của trung tâm.

1.5. Giao thức mạng

Giao thức mạng là một tập các quy tắc, quy ước để trao đổi thông tin giữa hai hệ thống máy tính hoặc hai thiết bị máy tính với nhau. Nói một cách hình thức thì giao thức mạng là một ngôn ngữ được các máy tính trong mạng sử dụng để trao đổi dữ liệu với nhau. Có nhiều loại giao thức được sử dụng trong mạng máy tính như: Apple Talk, DLC, NetBEUI,… nhưng hiện nay giao thức được sử dụng phổ biến nhất trong mạng máy tính là giao thức TCP/IP.

1.5.1. Giao thức TCP/IP

Giao thức TCP/IP được phát triển từ mạng ARPANET và Internet và được dùng như giao thức mạng và vận chuyển trên mạng Internet. TCP (Transmission Control Protocol) là giao thức thuộc tầng vận chuyển và IP (Internet Prorocol) là giao thức

thuộc tầng mạng của mô hình OSI. Họ giao thức TCP/IP hiện nay là giao thức được sử dụng rộng rãi nhất để liên kết các máy tính và các mạng.

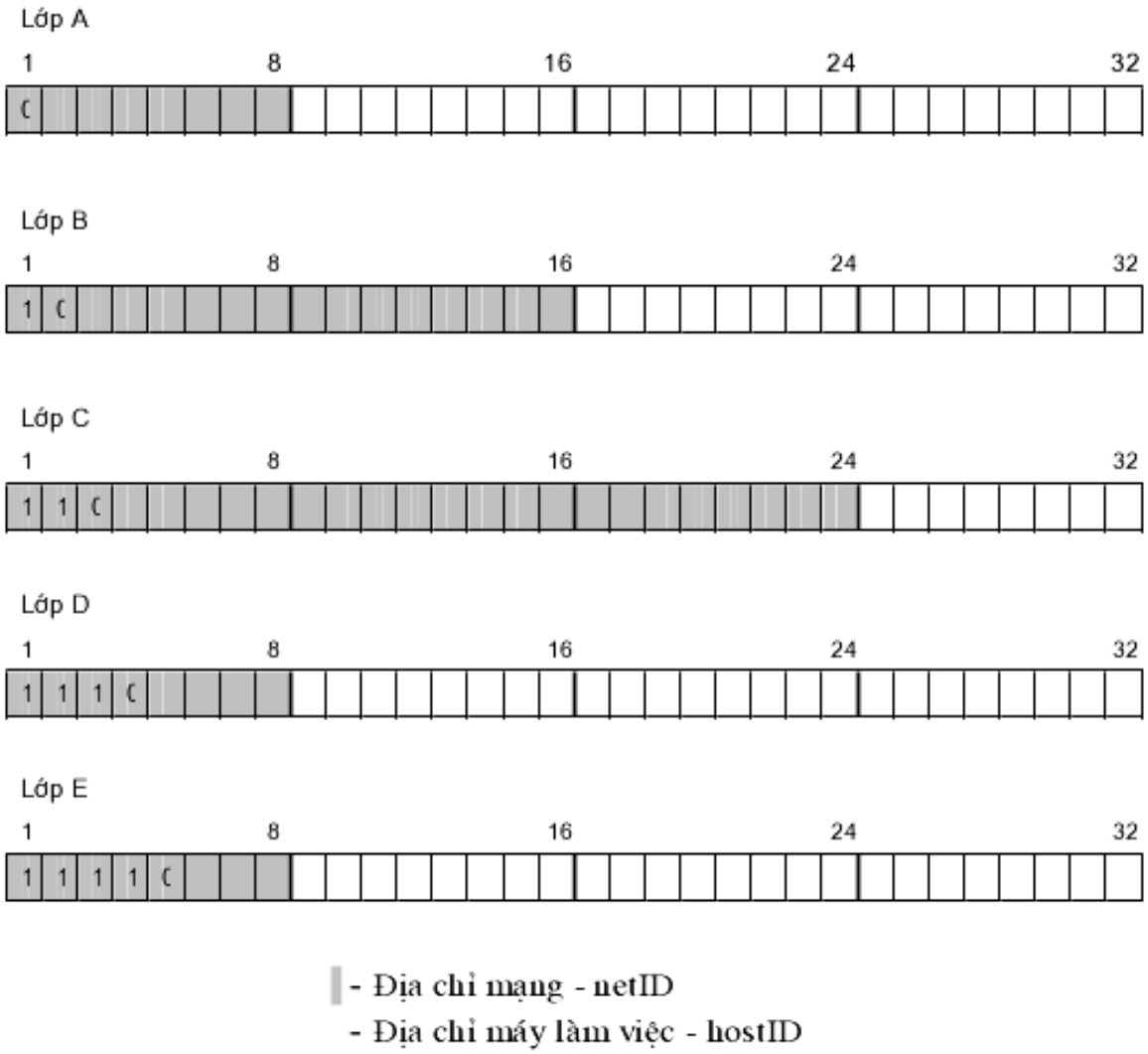
Hiện nay các máy tính của hầu hết các mạng có thể sử dụng giao thức TCP/IP để liên kết với nhau thông qua nhiều hệ thống mạng với kỹ thuật khác nhau. Giao thức TCP/IP thực chất là một họ giao thức cho phép các hệ thống mạng cùng làm việc với nhau thông qua việc cung cấp phương tiện truyền thông liên mạng.

1. Giao thức IP

Nhiệm vụ chính của giao thức IP là cung cấp khả năng kết nối các mạng con thành liên kết mạng để truyền dữ liệu, vai trò của IP là vai trò của giao thức tầng mạng trong mô hình OSI. Giao thức IP là một giao thức kiểu không hướng kết nối (connectionless), có nghĩa là không cần có giai đoạn thiết lập liên kết trước khi truyền dữ liệu.

Để định danh các trạm (host) trong liên mạng được người ta sử dụng địa chỉ IP có độ dài 32 bits. Mỗi giao diện trong một máy có hỗ trợ giao thức IP đều được gán một địa chỉ IP (một máy tính có thể gắn với nhiều mạng do vậy có thể có nhiều địa chỉ IP). Địa chỉ IP gồm 3 phần: bit định danh lớp mạng, địa chỉ mạng (netID) và địa chỉ máy (hostID). Mỗi địa chỉ IP được phân thành 4 vùng (mỗi vùng 1 byte), có thể biểu thị dưới dạng thập phân, bát phân, thập lục phân hay nhị phân. Cách viết phổ biến nhất là dùng ký pháp thập phân có dấu chấm (dotted decimal notation) để tách các vùng. Mục đích của địa chỉ IP là để định danh duy nhất cho một máy tính bất kỳ trên liên mạng.

Do tổ chức và độ lớn của mạng con (subnet) của liên mạng có thể khác nhau, người ta chia các địa chỉ IP thành 5 lớp, ký hiệu là A, B, C, D, và E. Trong lớp A, B, C chứa địa chỉ có thể gán được. Lớp D dành riêng cho lớp kỹ thuật multicasting. Lớp E được dành cho những ứng dụng trong tương lai.



Hình 1.5. Cấu trúc của các lớp địa chỉ IP

NetID dùng để nhận dạng từng mạng riêng biệt. Các mạng liên kết phải có địa chỉ mạng (netID) riêng cho mỗi mạng. Ở đây các bit đầu tiên của byte đầu tiên được dùng để định danh lớp địa chỉ (0 - lớp A, 10 - lớp B, 110 - lớp C, 1110 - lớp D và 11110 - lớp E).

Ở đây ta xét cấu trúc của các lớp địa chỉ có thể gán được là lớp A, B, C.

Phân lớp của địa chỉ IP như sau:

- *Mạng lớp A*: địa chỉ mạng (netID) là 1 byte và địa chỉ host (hostID) là 3 byte. Lớp A cho phép định dạng tới 126 mạng, tối đa hơn 16 triệu host trên mỗi mạng. Lớp này được dùng cho các mạng có số trạm cực lớn.
- *Mạng lớp B*: địa chỉ mạng (netID) là 2 byte và địa chỉ host (hostID) là 2 byte. Lớp B cho phép định danh tới 16382 mạng, với tối đa 65534 host trên mỗi mạng.

- *Mạng lớp C*: địa chỉ mạng (netID) là 3 byte và địa chỉ host (hostID) là 1 byte. Lớp C cho phép định danh tới 2 triệu mạng, với tối đa 254 host trên mỗi mạng. Lớp này được dùng cho các mạng có ít trạm.

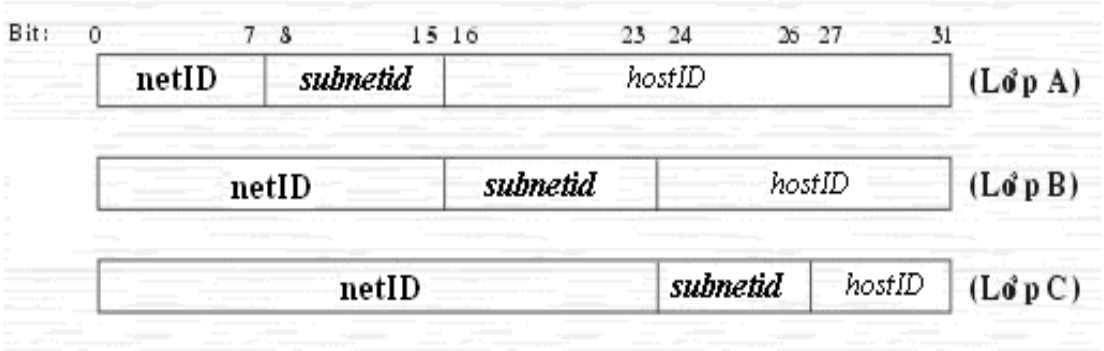
Lớp địa chỉ	Vùng địa chỉ lý thuyết	Bit nhận dạng	Số bit Net ID	Số mạng tối đa	Số máy tối đa
A	Từ 0.0.0.0 đến 127.0.0.0	0	7	126	16777214
B	Từ 128.0.0.0 đến 191.0.0.0	10	14	16382	65534
C	Từ 192.0.0.0 đến 223.0.0.0	110	21	2097150	254
D	Từ 224.0.0.0 đến 240.0.0.0	1110			
E	Từ 241.0.0.0 đến 255.0.0.0	11110			

Hình 1.6. Bảng phân lớp địa chỉ IP

Một số địa chỉ có tính chất đặc biệt: một địa chỉ có hostID = 0 được dùng để hướng tới mạng định danh bởi vùng netID. Ngược lại, một địa chỉ có vùng hostID gồm toàn số 1 dùng để hướng tới tất cả các host nối vào mạng netID, và nếu vùng netID cũng gồm toàn số 1 thì nó hướng tới tất cả các host liên mạng.

Cần lưu ý rằng địa chỉ IP được dùng để định danh các host và mạng ở tầng OSI, và chúng không phải là các địa chỉ vật lý (hay địa chỉ MAC) của các trạm trên đó một mạng cục bộ (Ethernet, Token Ring).

Trong nhiều trường hợp, một mạng có thể được chia làm nhiều mạng con (subnet), lúc đó có thể đưa thêm các vùng subnetid để định danh các mạng con. Vùng subnetid được lấy từ vùng hostID, cụ thể đối với lớp A, B, C như ví dụ sau:



Hình 1.7. Ví dụ địa chỉ IP khi bổ sung subnetid

2. Giao thức TCP

TCP là một giao thức hướng kết nối, có cung cấp một đường truyền dữ liệu tin cậy giữa hai máy tính. Tính tin cậy của đường truyền được thể hiện ở hai đặc điểm sau:

- Mọi gói tin cần gửi sẽ đến được đích. Để làm điều này thì mỗi lần phía gửi sau khi gửi xong một gói tin nó sẽ chờ nhận một biên nhận từ bên nhận rằng đã nhận được đúng gói tin. Nếu sau một khoảng thời gian mà phía gửi không nhận được thông tin xác nhận phản hồi thì nó sẽ phát lại gói tin. Việc phát lại sẽ được tiến hành cho đến khi việc truyền tin thành công, tuy nhiên sau một số lần phát lại max nào đó mà vẫn chưa thành công thì phía gửi có thể suy ra là không thể truyền tin được và sẽ dừng việc phát tin.

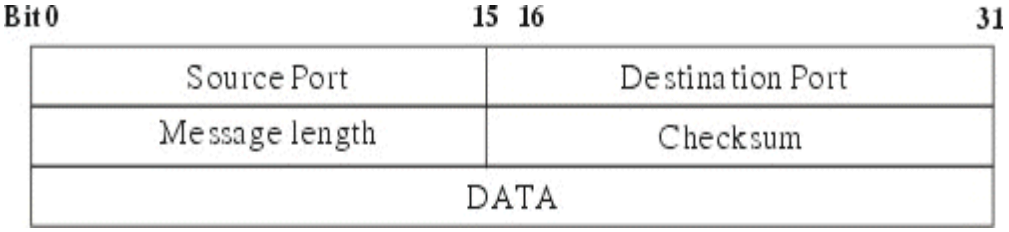
- Các gói tin sẽ được trình ứng dụng nhận được theo đúng thứ tự như chúng được gửi đi. Bởi các gói tin có thể được dẫn đi trên mạng theo nhiều con đường khác nhau trước khi tới đích nên thứ tự khi tới đích của chúng có thể không giống như khi chúng được phát. Do đó để đảm bảo có thể sắp xếp lại các gói tin một cách đúng đắn như ở phía gửi, giao thức TCP sẽ đánh số thứ tự cho từng gói tin trong cả khối tin chung được phát, nhờ vậy bên nhận có thể sắp xếp lại các gói tin theo đúng thứ tự ban đầu của chúng.

Như vậy có thể thấy TCP cung cấp cho chúng ta một kênh truyền thông điểm - điểm phục vụ cho các ứng dụng đòi hỏi giao tiếp tin cậy như HTTP (*Hypertext Tranfer Protocol*), FPT (*File Tranfer Protocol*),… Các ứng dụng này đỏi hỏi một kênh giao tiếp tin cậy bởi thứ tự dữ liệu được gửi và nhận là yếu tố quyết định đến sự thành công hay thất bại của chúng.

1.5.2 Giao thức UDP

UDP (*User Datagram Protocol*) là giao thức không hướng kết nối, được sử dụng thay thế cho TCP theo yêu cầu của từng ứng dụng. Khác với TCP, UDP không có các chức năng thiết lập và kết thúc kết nối. Tương tự như IP, nó cũng không cung cấp cơ chế báo nhận (*acknowledgment*), không sắp xếp tuần tự các gói tin (*datagram*) đến và có thể dẫn đến tình trạng mất hoặc trùng dữ liệu mà không có cơ chế thông báo lỗi cho người gửi. Qua đó ta thấy UDP cung cấp các dịch vụ vận chuyển không tin cậy như trong TCP.

Khuôn dạng UDP datagram được mô tả với các vùng tham số đơn giản hơn nhiều so với TCP segment.



Hình 1.8. Khuôn dạng của gói tin UDP

1.6. Các mô hình hoạt động của mạng máy tính

Mô hình hoạt động của mạng máy tính có hai loại:

- Mô hình mạng hoạt động theo dạng peer to peer
- Mô hình mạng hoạt động theo dạng clients/server

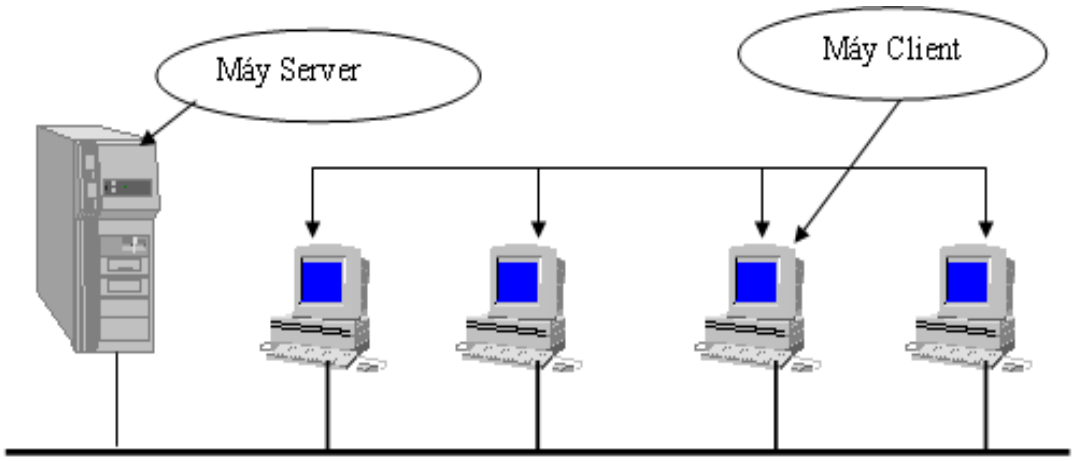
1.6.1. Mô hình mạng hoạt động theo dạng peer to peer

Không tồn tại bất kỳ máy chuyên dụng hoặc cấu trúc phân cấp giữa các máy tính, mọi máy tính đều bình đẳng và có vai trò như nhau. Thông thường mỗi máy tính hoạt động với cả vai trò máy khách và máy phục vụ, vì vậy không có máy nào được chỉ định quản lý toàn mạng. Người dùng ở từng máy tự quyết định dữ liệu nào trên máy của mình sẽ được chia sẻ để dùng chung trên mạng.

Hình 1.9. Mô hình mạng hoạt động theo dạng peer to peer

1.6.2. Mô hình mạng hoạt động theo dạng clients/ server

Trong mạng hoạt động theo mô hình clients/server có một hoặc nhiều máy có nhiệm vụ cung cấp một số dịch vụ cho các máy khác ở trong mạng, các máy này được gọi là server còn các máy tính được phục vụ gọi là máy clients.



Hình 1.10. Mô hình mạng hoạt động theo dạng clients/server

Đây là mô hình tổng quát, trên thực tế server có thể được nối với nhiều server khác để tăng hiệu quả làm việc. Khi nhận được yêu cầu từ clients, server có thể xử lý yêu cầu đó hoặc gửi tiếp yêu cầu vừa nhận được cho một server khác.

Máy server sẽ thi hành các nhiệm vụ do máy clients yêu cầu. Có rất nhiều dịch vụ trên mạng hoạt động theo nguyên lý nhận các yêu cầu từ clients sau đó xử lý và trả lại các kết quả cho clients yêu cầu.

CHƯƠNG 2: SƠ LƯỢC VỀ NGÔN NGỮ LẬP TRÌNH JAVA

2.1. Giới thiệu

Java là một ngôn ngữ lập trình được Sun Microsystems giới thiệu vào tháng 6 năm 1995. Từ đó, nó đã trở thành một công cụ lập trình của các lập trình viên chuyên nghiệp. Java được xây dựng trên nền tảng của C và C++, do vậy nó sử dụng các cú pháp của C và các đặc trưng hướng đối tượng của C++.

Vào năm 1991, một nhóm các kỹ sư của Sun Microsystems có ý định thiết kế một ngôn ngữ lập trình để điều khiển các thiết bị điện tử như tivi, máy giặt, lò nướng,… Mặc dù C và C++ có khả năng làm việc này nhưng trình biên dịch lại phụ thuộc vào từng loại CPU.

Trình biên dịch thường phải tốn nhiều thời gian để xây dựng nên rất đắt, vì vậy để mỗi loại CPU có một trình biên dịch riêng là rất tốn kém. Do đó nhu cầu thực tế đòi hỏi một ngôn ngữ chạy nhanh, gọn, hiệu quả và độc lập thiết bị tức là có thể chạy trên nhiều loại CPU khác nhau, dưới các môi trường khác nhau. “Oak” đã ra đời và vào năm 1995 được đổi tên thành Java. Mặc dù mục tiêu ban đầu không phải cho Internet nhưng do đặc trưng không phụ thuộc thiết bị nên Java đã trở thành ngôn ngữ lập trình cho Internet.

2.2. Một số tính chất của ngôn ngữ Java

Java là ngôn ngữ lập trình được phát triển từ ngôn ngữ lập trình C/C++. Nó kế thừa, phát huy các thế mạnh của ngôn ngữ C/C++ và lược bỏ đi các cú pháp phức tạp của C/C++. Ngôn ngữ lập trình Java có một số đặc trưng tiêu biểu: đơn giản, hướng đối tượng, độc lập phần cứng và hệ điều hành, mạnh mẽ, bảo mật, phân tán, đa luồng và linh động.

2.2.1. Đơn giản

Những người thiết kế mong muốn phát triển một ngôn ngữ dễ học và quen thuộc với đa số người lập trình. Do vậy Java loại bỏ các đặc trưng phức tạp của C và C++ như:

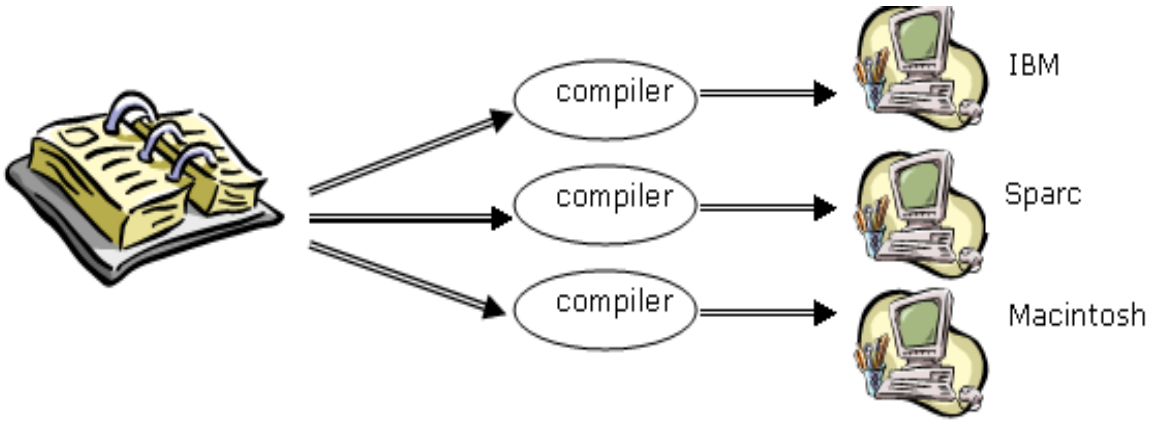
- Loại bỏ thao tác con trỏ, thao tác định nghĩa chồng toán tử
- Không cho phép đa kế thừa mà sử dụng các giao diện
- Không sử dụng lệnh “`goto`” cũng như file header (`.h`)
- Loại bỏ cấu trúc “`struct`” và “`union`”

2.2.2. Hướng đối tượng

Java là ngôn ngữ lập trình thuần hướng đối tượng, mọi chương trình viết trên Java đều phải được xây dựng trên các đối tượng. Nếu trong C/C++ ta có thể tạo ra các hàm (chương trình con không gắn với đối tượng nào) thì trong Java ta chỉ có thể tạo ra các phương thức (chương trình con gắn liền với một lớp cụ thể). Trong Java không cho phép các đối tượng có tính năng đa kế thừa mà được thay thế bằng các giao diện (`interface`)

2.2.3. Độc lập phần cứng và hệ điều hành

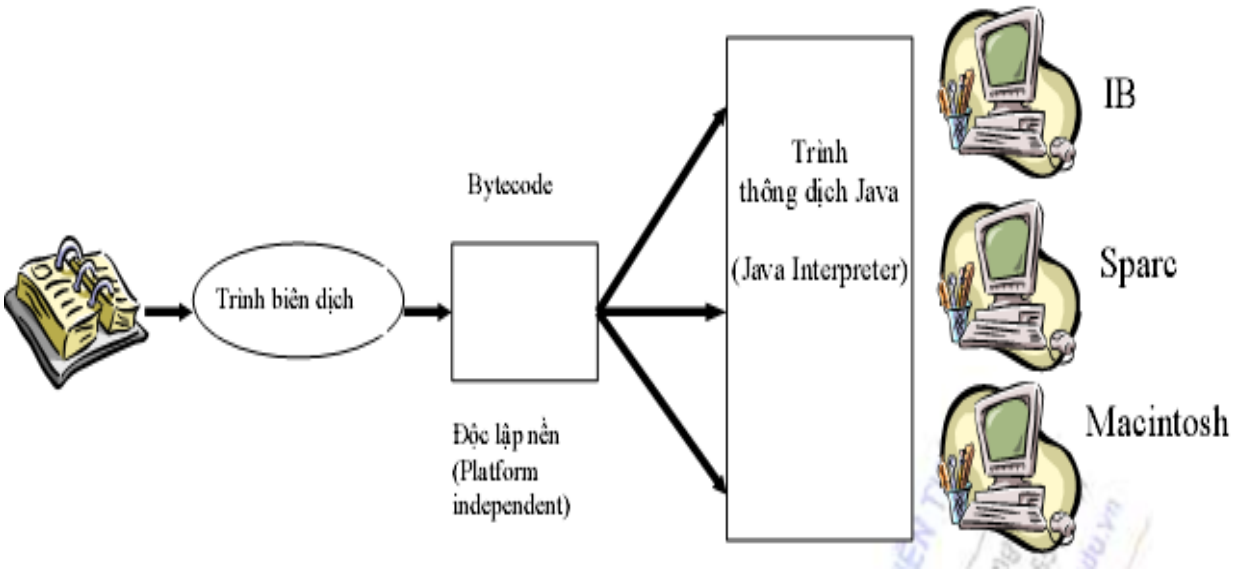
Đối với các ngôn ngữ lập trình truyền thống như C/C++, phương pháp biên dịch được thực hiện như sau :



Hình 2.1. Cách biên dịch chương trình truyền thống

Với mỗi nền phần cứng khác nhau, có một trình biên dịch khác nhau để biên dịch mã nguồn chương trình cho phù hợp với nền phần cứng ấy. Do vậy, khi chạy trên một nền phần cứng khác bắt buộc phải biên dịch lại mã nguồn.

Đối với các chương trình viết bằng Java, trình biên dịch Javac sẽ biên dịch mã nguồn thành dạng bytecode. Sau đó, khi chạy chương trình trên các nền phần cứng khác nhau, máy ảo Java dùng trình thông dịch Java để chuyển mã bytecode thành dạng chạy được trên các nền phần cứng tương ứng. Do vậy, khi thay đổi nền phần cứng, không phải biên dịch lại mã nguồn Java.



Hình 2.2. Cơ chế dịch chương trình Java

2.2.4. Mạnh mẽ

Java là ngôn ngữ yêu cầu chặt chẽ về kiểu dữ liệu.

- Kiểu dữ liệu phải khai báo tường minh.

- Java không sử dụng con trỏ và các phép toán con trỏ.

- Java kiểm tra tất cả các truy nhập đến mảng, chuỗi khi thực thi để đảm bảo rằng các truy nhập đó không ra ngoài giới hạn kích thước

- Trong các môi trường lập trình truyền thống, lập trình viên phải tự mình cấp phát bộ nhớ, trước khi chương trình kết thúc thì phải tự giải phóng bộ nhớ đã cấp. Vấn đề có thể nảy sinh khi lập trình viên quên giải phóng bộ nhớ đã xin cấp trước đó. Trong chương trình Java, lập trình viên không phải bận tâm đến việc cấp phát bộ nhớ. Quá trình cấp phát, giải phóng được thực hiện tự động, nhờ dịch vụ thu nhặt những đối tượng không còn sử dụng nữa (garbage collection).

- Cơ chế bẫy lỗi của Java giúp đơn giản hóa qúa trình xử lý lỗi và hồi phục sau lỗi.

2.2.5. Bảo mật

Java cung cấp một môi trường quản lý thực thi chương trình với nhiều mức để kiểm soát tính an toàn:

- Ở mức thứ nhất, dữ liệu và các phương thức được đóng gói bên trong lớp. Chúng chỉ được truy xuất thông qua các giao diện mà lớp cung cấp.

- Ở mức thứ hai, trình biên dịch kiểm soát để đảm bảo mã là an toàn, và tuân theo các nguyên tắc của Java.

- Mức thứ ba được đảm bảo bởi trình thông dịch; chúng kiểm soát xem bytecode có đảm bảo các quy tắc an toàn trước khi thực thi không.

- Mức thứ tư kiểm soát việc nạp các lớp vào bộ nhớ để giám sát việc vi phạm giới hạn truy xuất trước khi nạp vào hệ thống.

2.2.6. Phân tán

Java được thiết kế để hỗ trợ các ứng dụng chạy trên mạng bằng các lớp mạng (`java.net`). Hơn nữa, Java hỗ trợ nhiều nền chạy khác nhau nên chúng được sử dụng rộng rãi như là công cụ phát triển trên Internet - nơi sử dụng nhiều nền khác nhau.

2.2.7. Đa luồng

Chương trình Java cung cấp giải pháp đa luồng (Multithreading) để thực thi các công việc đồng thời. Chúng cũng cung cấp giải pháp đồng bộ giữa các luồng. Đặc tính hỗ trợ đa luồng này cho phép xây dựng các ứng dụng trên mạng chạy hiệu quả.

2.2.8. Linh động

Java được thiết kế như một ngôn ngữ động để đáp ứng cho những môi trường mở. Các chương trình Java chứa rất nhiều thông tin thực thi nhằm kiểm soát và truy nhập đối tượng lúc chạy. Điều này cho phép khả năng liên kết mã động.

2.3. Các dạng chương trình ứng dụng của Java

2.3.1. Chương trình ứng dụng dạng độc lập (Application)

Chương trình ứng dụng dạng độc lập là một chương trình nguồn mà sau khi dịch có thể thực hiện trực tiếp. Chương trình ứng dụng dạng độc lập trong Java bắt đầu thực hiện và kết thúc ở phương thức `main()`, giống như hàm `main()` trong chương trình C/C++.

Khi xây dựng một ứng dụng độc lập cần lưu ý:

1. Tạo lập một lớp được định nghĩa bởi người sử dụng có phương thức `main()` gọi là lớp chính và bảo đảm nó được định nghĩa đúng theo đúng nguyên mẫu được quy định bởi Java.

2. Kiểm tra xem liệu tệp chương trình có tên trùng với tên của lớp chính và đuôi là "`.java`" hay không.

3. Dịch tệp chương trình nguồn "`.java`" để tạo ra các tệp mã bytecode có đuôi "`.class`" tương ứng.

4. Sử dụng chương trình thông dịch của Java để chạy chương trình đã dịch.

2.3.2. Chương trình ứng dụng dạng nhúng (Applet)

Applet là loại chương trình Java đặc biệt mà khi thực hiện mã lệnh của chúng phải được nhúng trong vào một trang web (các file có đuôi HTM hoặc HTML), các thẻ HTML sẽ được trình duyệt Web thực thi (như Netscape hoặc Internet Explorer) còn đoạn mã lệnh của Applet sẽ được máy ảo Java nhúng trong trình duyệt Web thực thi. Cũng có thể dùng trình Appletviewer của JDK để thực thi một Applet.

Một chương trình dạng Applet bao gồm hai tệp: "`.java`" và "`.html`"

Chu trình hoạt động cuả Applet:

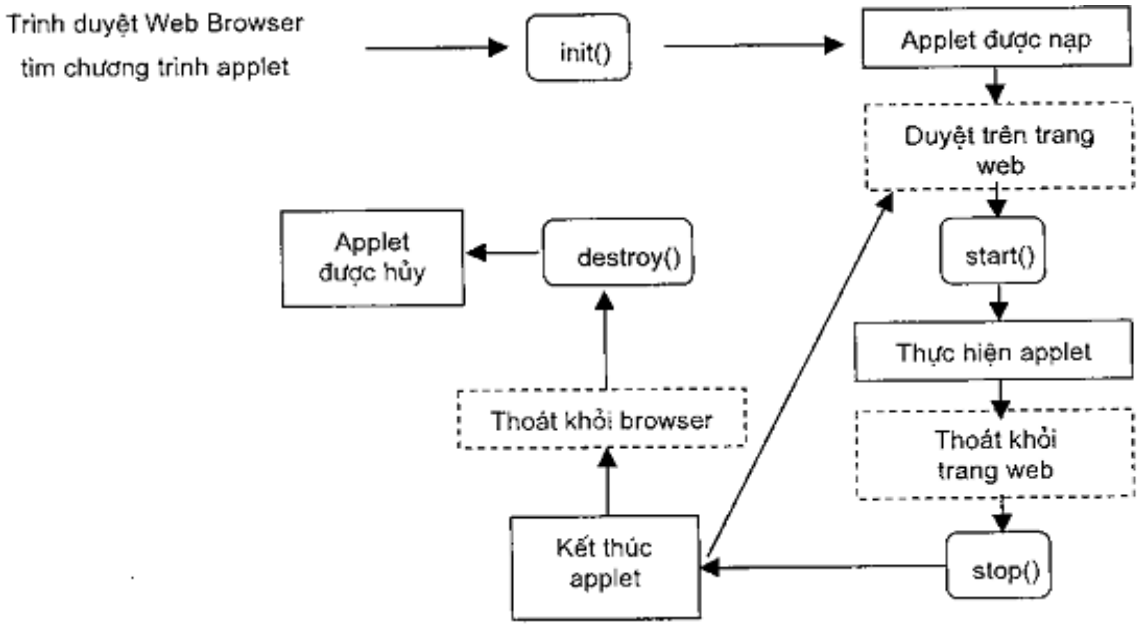
Chương trình ứng dụng Applet được thực hiện như sau:

- Khi một applet được nạp và chạy bởi Web Browser thì nó sẽ gửi thông điệp `init()` cùng với các dữ liệu, kích thước của Window để chương trình Applet khởi động.

- Khi bắt đầu thực hiện, Web Browser thông báo cho Applet bắt đầu bằng cách gọi phương thức `start()`.

- Khi rời khỏi trang Web có chứa Applet thì chương trình Applet này nhận được thông điệp `stop()` để dừng chương trình.

Hoạt động của chương trình Applet được mô tả như hình dưới



Hình 2.3. Chu trình hoạt động của Applet

Trong đó:

- *init():* phương thức này được gọi khi Applet được nạp lần đầu và được xem như là toán tử tạo lập cho Applet.
- *start():* được gọi khi Applet bắt đầu thực hiện, xuất hiện khi:
  + Applet được nạp xuống
  + Applet được duyệt lại
- *stop():* được gọi khi Applet dừng thực hiện, nhưng chưa bị loại bỏ khỏi bộ nhớ.
- *destroy():* được gọi ngay trước khi Applet kết thúc, khi trình duyệt Web được đóng lại và Applet bị loại bỏ khỏi bộ nhớ.

2.3.3. Chương trình ứng dụng dạng lai ghép

Java cho phép xây dựng một chương trình có thể chạy được cả ở Web Browser (Applet) cũng như một ứng dụng dạng độc lập (Application), để xây dựng được một chương trình như thế phải:

- Định nghĩa lớp ứng dụng kế thừa từ lớp `Applet`
- Trong lớp ứng dụng phải có hàm `main()`

2.4. Cấu trúc của tệp chương trình Java

Tệp chương trình Java có thể có các phần được đặc tả như sau:

- Định nghĩa một gói là tùy chọn thông qua định danh của gói (`package`). Tất cả các lớp, các `interface` được định nghĩa trong tệp chứa gói này đều thuộc gói đó. Nếu bỏ qua định nghĩa gói thì các định nghĩa ở tệp này sẽ thuộc vào gói mặc định.
- Một số lệnh nhập `import`.
- Một số định nghĩa lớp và `interface` có thể định nghĩa theo thứ tự bất kỳ, trong đó thường có một lớp `public`.

Như vậy, cấu trúc của một tệp chương trình Java có thể khái quát như sau:

```
// Filename: New.java
    // Phần 1: tùy chọn
        // Định nghĩa gói
        package Têngói;
    // Phần 2: 0 hoặc nhiều hơn
        // các gói cần sử dụng
        import java.io.*;
```

```
// Phần 3: 0 hoặc nhiều hơn
        // Định nghĩa các lớp và các interface
        public class New{...}
        class C1 {...}
        interface I1 {...}
        // ...
        class Cn {...}
        interface Im {...}
```

CHƯƠNG 3 : LẬP TRÌNH SOCKET TCP

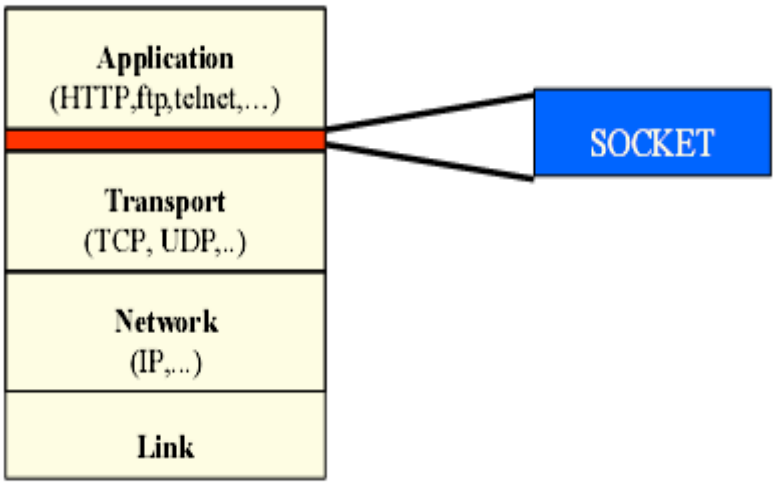
3.1. Định nghĩa

Có nhiều định nghĩa khác nhau về **socket** tùy theo cách nhìn của người sử dụng:

Một cách tổng quát nhất có thể định nghĩa: một socket là một điểm cuối trong một kết nối giữa hai chương trình đang chạy trên mạng.

Nhìn trên quan điểm của người phát triển ứng dụng người ta có thể định nghĩa socket là một phương pháp thiết lập kết nối truyền thông giữa một chương trình yêu cầu dịch vụ (được gán nhãn là clients) và một chương trình cung cấp dịch vụ (được gán nhãn là server) trên mạng hoặc trên cùng một máy tính.

Đối với người lập trình, họ nhìn nhận socket như một giao diện nằm giữa tầng ứng dụng và tầng khác trong mô hình mạng OSI, có nhiệm vụ thực hiện việc giao tiếp giữa chương trình ứng dụng với các tầng bên dưới của mạng.



Hình 3.1. Mô hình OSI dạng rút gọn

Tuy nhiên, các lập trình viên hiện nay gần như luôn luôn bị ngăn cản tạo socket riêng bằng cách thủ công, bởi dù bạn dùng Java hay PHP,…, có thể bạn sẽ không bao giờ mở được cổng một cách tường minh. Thay vào đó các lập trình viên sẽ dùng thư viện socket được hỗ trợ sẵn bởi các ngôn ngữ lập trình. Như vậy, các socket vẫn tồn tại để kết nối các ứng dụng của người dùng, nhưng các chi tiết của socket được ẩn trong những lớp sâu hơn để mọi người không phải động chạm đến.

Do *socket* là một thực thể phần mềm có chức năng nhận hoặc gửi dữ liệu đi trên kết nối giữa hai ứng dụng mạng nên khi cần sử dụng socket thì ứng dụng sẽ tạo ra *socket* để dùng, khi không cần sử dụng nữa thì có thể huỷ bỏ socket.

Một *socket* được định danh bằng một cặp giá trị:

- Địa chỉ IP của máy tính có chương trình ứng dụng đã tạo ra socket
- Số hiệu cổng (port) mà socket dùng để nhận/gửi dữ liệu.

Khái niệm cổng: Cổng thực chất là số hiệu của một chương trình ứng dụng đang chạy trên một máy tính. Để hệ thống có thể theo dõi được các chương trình ứng dụng đang chạy trên máy tính, hệ điều hành sẽ gán cho mỗi ứng dụng đó một con số (16bits) trong khoảng từ 0 đến 65535. Trong thực tế thì các số hiệu cổng từ 0 đến 1023 (gồm 1024 cổng) đã được dùng cho các dịch vụ nổi tiếng :

Số hiệu cổng	Quá trình hệ thống
7	Dịch vụ Echo
21	Dịch vụ FTP
23	Dịch vụ Telnet
25	Dịch vụ E-mail (SMTP)
80	Dịch vụ Web (HTTP)
110	Dịch vụ E-mail (POP)

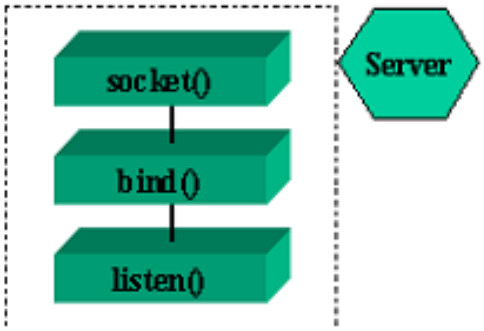
Hình 3.2. Số hiệu cổng của một số dịch vụ nổi tiếng

Nếu chúng ta không phải là người quản trị thì nên dùng từ cổng 1024 trở lên.

Vậy **socket** = Địa chỉ **IP** + Số hiệu **Port**

3.2. Mô hình clients/server sử dụng socket ở chế độ hướng kết nối TCP

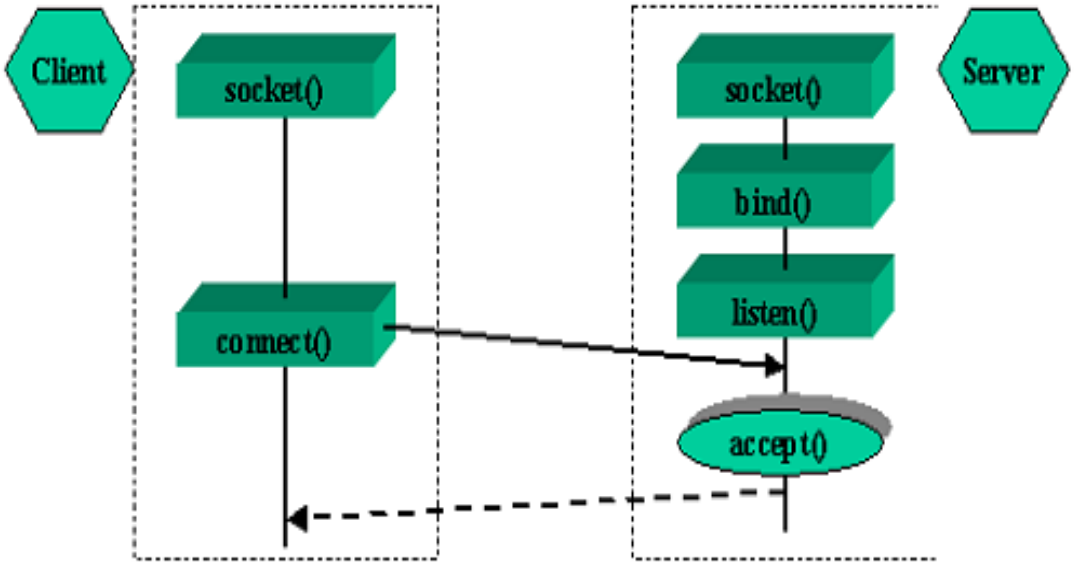
Giai đoạn 1: Server tạo socket, gán số hiệu cổng và lắng nghe yêu cầu kết nối.



- *socket()*: Server yêu cầu tạo một socket để có thể sử dụng các dịch vụ của tầng vận chuyển.
- *bind():* Server yêu cầu gán số hiệu cổng (port) cho socket.
- *listen()*: Server lắng nghe các yêu cầu kết nối từ clients trên cổng đã được gán.

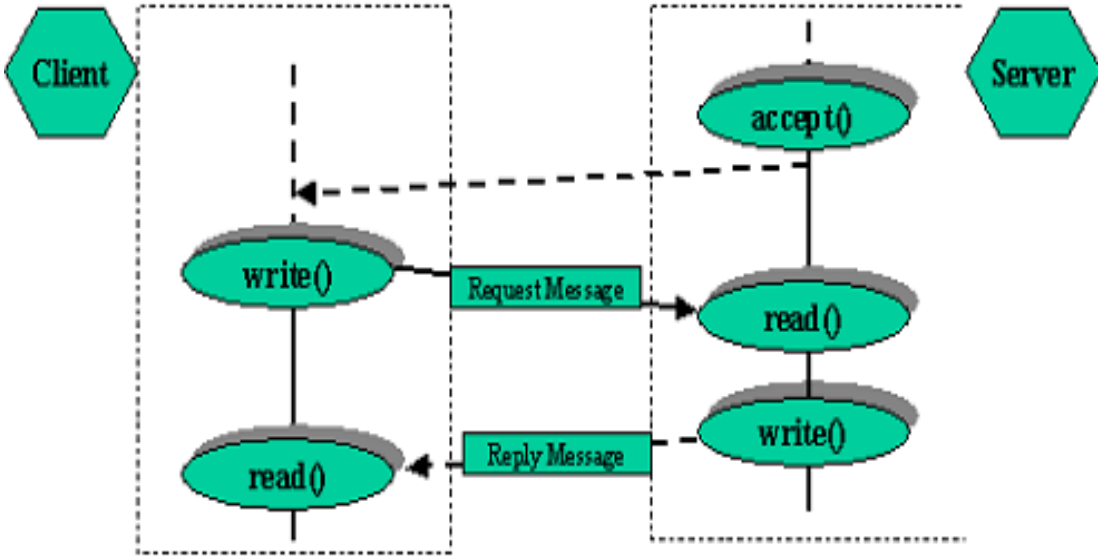
Server sẵn sàng phục vụ clients.

Giai đoạn 2: Clients tạo socket, yêu cầu thiết lập một kết nối tới server.



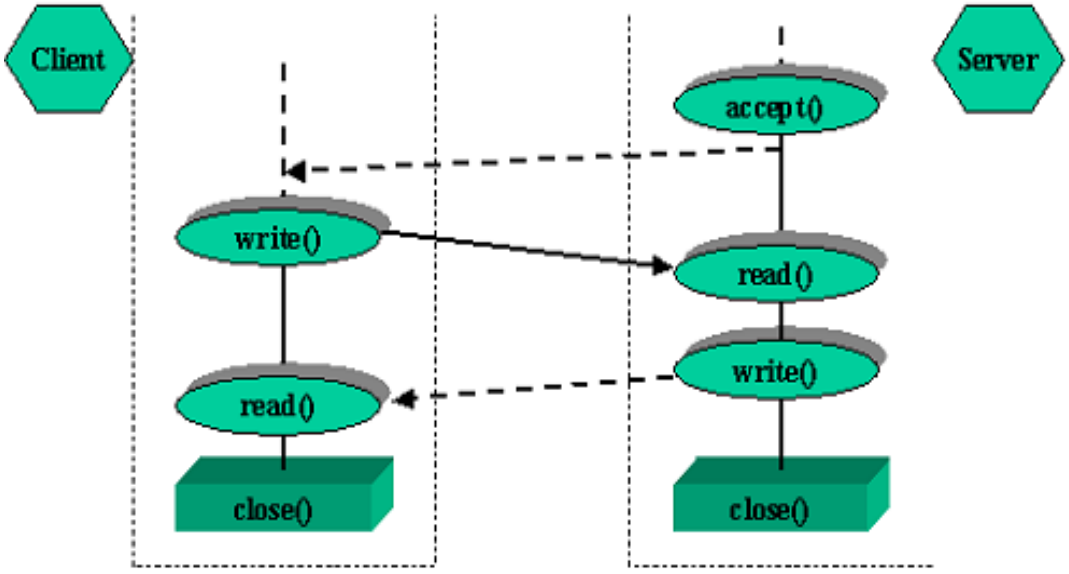
- *socket():* Clients yêu cầu tạo một socket để có thể sử dụng các dịch vụ của tầng vận chuyển, thông thường hệ thống tự động gán một số hiệu cổng còn rảnh cho socket của clients.
- *connect():* Clients gửi yêu cầu nối kết đến server có địa chỉ IP và Port xác định.
- *accept()*: Server chấp nhận kết nối của clients, khi đó một kênh giao tiếp ảo được hình thành, clients và server có thể trao đổi thông tin với nhau thông qua kênh ảo này.

Giai đoạn 3: Trao đổi thông tin giữa clients và server



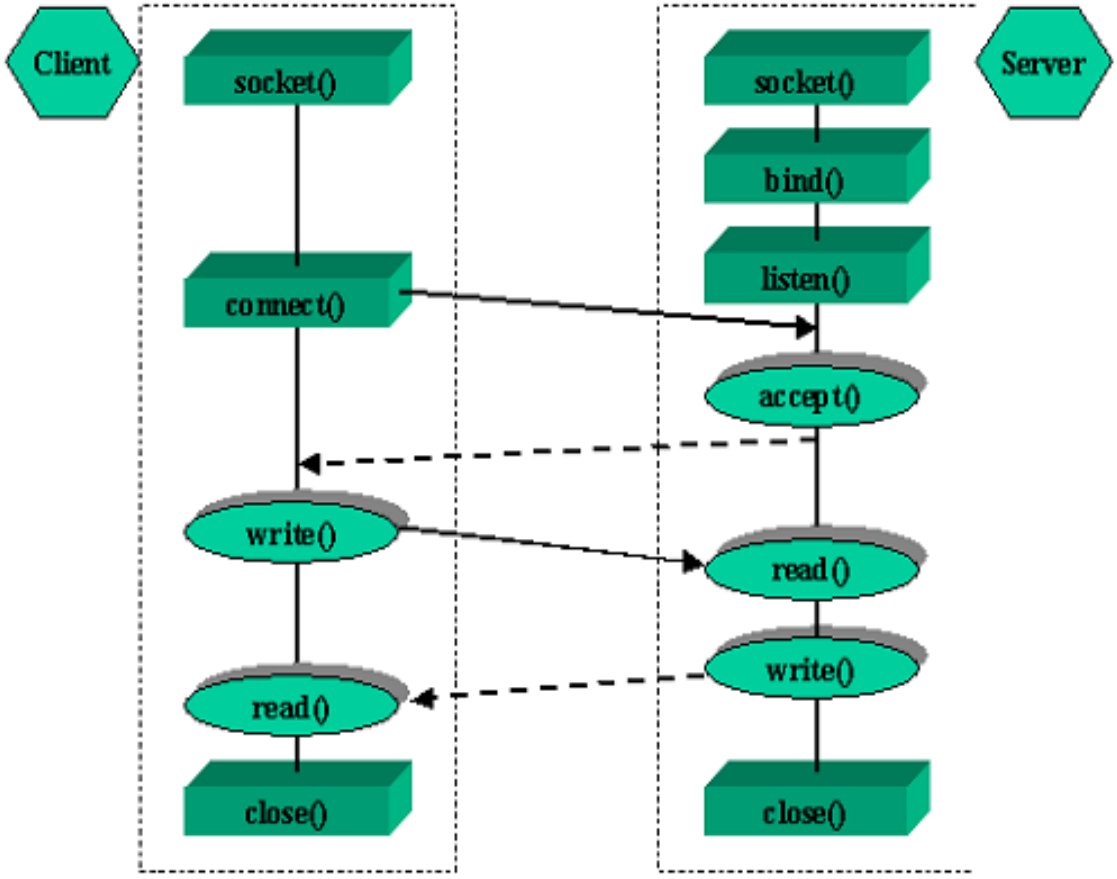
- Sau khi chấp nhận yêu cầu kết nối, thông thường server thực hiện lệnh *read()* và nghẽn cho đến khi có thông điệp yêu cầu (*Request Message*) từ clients gửi đến.
- Server phân tích và thực thi yêu cầu, kết quả sẽ được gửi về clients bằng lệnh *write()*.
- Sau khi gửi yêu cầu bằng lệnh *write()*, clients chờ nhận thông điệp kết quả (*Reply Message*) từ server bằng lệnh *read()*.

Giai đoạn 4 : Kết thúc phiên làm việc



- Các câu lệnh *read()*, *write()* có thể được thực hiện nhiều lần (ký hiệu bằng hình ellipse).
- Kênh ảo sẽ bị xóa khi server hoặc clients đóng socket bằng lệnh *close()*.

Như vậy toàn bộ quá trình diễn ra như sau :



3.3. Lập trình Socket TCP trong Java

Java hỗ trợ lập trình mạng thông qua các lớp trong gói `java.net`. Một số lớp tiêu biểu được dùng cho lập trình clients/server sử dụng socket làm phương tiện giao tiếp như:

- `InetAddress`: Lớp này biểu diễn địa chỉ Internet, quan trọng nhất là hai phương thức `getHostName()` và `getAddress()` dùng để chuyển đổi giữa địa chỉ IP và tên máy tính.
- `Socket`: Hỗ trợ các phương thức liên quan đến socket cho chương trình clients ở chế độ hướng kết nối.
- `ServerSocket`: Hỗ trợ các phương thức liên quan đến socket cho chương trình server ở chế độ hướng kết nối.
- `DatagramSocket`: Hỗ trợ các phương thức liên quan đến socket ở chế độ không hướng kết nối cho cả clients và server.

• `DatagramPacket`: Lớp cài đặt gói tin dạng thư tín người dùng trong giao tiếp giữa clients và server ở chế độ không hướng kết nối.

3.3.1. Xây dựng chương trình clients ở chế độ hướng kết nối

Các bước tổng quát:

1. Mở một socket kết nối đến server đã biết địa chỉ IP (hay tên miền) và số hiệu cổng
2. Lấy `InputStream` và `OutputStream` gán với socket
3. Tham khảo protocol của dịch vụ để định dạng đúng dữ liệu trao đổi với server
4. Trao đổi dữ liệu với server nhờ vào các `InputStream` và `OutputStream`
5. Đóng socket trước khi kết thúc chương trình

Lớp **`java.net.Socket:`** lớp socket hỗ trợ các phương thức cần thiết để xây dựng các chương trình clients sử dụng ở chế độ hướng kết nối. Dưới đây là một số phương thức thường dùng để xây dựng clients

- public Socket(String HostName, int PortNumber) throws IOException: phương thức này dùng để kết nối đến một server có tên là HostName, cổng là PortNumber. Nếu kết nối thành công, một kênh ảo sẽ được hình thành giữa clients và server.

+ HostName : địa chỉ IP hoặc tên logic theo dạng tên miền
+ PortNumber : có giá trị từ 0 đến 65535

Ví dụ: mở socket và kết nối đến Web Server của khoa công nghệ thông tin, đại học Cần Thơ:

```
Socket s = new Socket (www.cit.ctu.edu.vn,80);
```

hoặc `Socket s = new Socket("203.162.36.149",80);`

- public InputStream getInputStream(): phương thức này trả về `InputStream` nối với socket. Chương trình clients dùng `InputStream` này để nhận dữ liệu từ server gửi về.

Ví dụ : lấy `InputStream` của socket s:

```
InputStream is = s.getInputStream();
```

- public OutputStream getOutputStream(): phương thức này trả về `OutputStream` nối với socket. Chương trình clients dùng `OutputStream` này để gửi dữ liệu cho server.

Ví dụ: Lấy `OutputStream` của socket s :

```
OutputStream os = s.getOutputStream();
```

- ***public close():*** phương thức này sẽ đóng socket lại, giải phóng kênh ảo, xoá kết nối giữa clients và server.

Ví dụ : Đóng socket s :

```
s.close();
```

3.3.2. Xây dựng chương trình server ở chế độ hướng kết nối

Lớp **`java.net.ServerSocket`**: hỗ trợ các phương thức cần thiết để xây dựng các chương trình server sử dụng socket ở chế độ hướng kết nối. Dưới đây là một số phương thức thường dùng để xây dựng server

- ***public ServerSocket(int PortNumber :*** phương thức này tạo một socket với số hiệu cổng là PortNumber mà sau đó server sẽ lắng nghe trên cổng này

Ví dụ : tạo socket cho server với số hiệu cổng là 7 :

```
ServerSocket ss = new ServerSocket(7);
```

- ***public Socket accept()*** : phương thức này lắng nghe yêu cầu kết nối của clients. Đây là một phương thức hoạt động ở chế độ nghẽn; nó sẽ bị nghẽn cho đến khi có một yêu cầu kết nối của clients gửi đến. Khi có yêu cầu kết nối của clients gửi đến, nó sẽ chấp nhận yêu cầu kết nối, trả về một socket là một đầu của kênh giao tiếp ảo giữa server và clients yêu cầu kết nối.

Ví dụ: Socket ss chờ nhận yêu cầu nối kết :

```
Socket s = ss.accept();
```

Server sau đó sẽ lấy `InputStream` và `OutputStream` của socket mới s để giao tiếp với clients.

Xây dựng chương trình server phục vụ tuần tự

Một server có thể được cài đặt để phục vụ clients theo hai cách: phục vụ tuần tự hoặc phục vụ song song.

Trong chế độ phục vụ tuần tự, tại một thời điểm server chỉ chấp nhận một yêu cầu kết nối, các yêu cầu kết nối của clients khác đều không được đáp ứng (đưa vào hàng đợi).

Ngược lại, trong chế độ phục vụ song song, tại một thời điểm server chấp nhận nhiều yêu cầu kết nối và phục vụ nhiều clients cùng lúc

Trong phần này, ta sẽ tìm hiểu về chế độ phục vụ tuần tự của server, còn chương tiếp sẽ tìm hiểu cụ thể về chế độ phục vụ song song (sau khi đã tìm hiểu về Thread).

Các bước tổng quát của một server phục vụ tuần tự :

- Tạo socket và gán số hiệu cổng cho server
- Lắng nghe yêu cầu kết nối
- Với một yêu cầu kết nối được chấp nhận thực hiện các bước sau:

+ lấy `InputStream` và `OutputStream` gắn với socket của kênh ảo vừa được hình thành

+ lặp lại công việc sau:

- ✔ Chờ nhận các yêu cầu (công việc)
- ✔ Phân tích và thực hiện yêu cầu
- ✔ Tạo thông điệp trả lời
- ✔ Gửi thông điệp trả lời về clients
- ✔ Nếu không còn yêu cầu hoặc clients kết thúc, đóng socket và quay lại bước2

CHƯƠNG 4: LUỒNG TRONG JAVA

4.1. Khái niệm luồng

- Luồng là một cách thông dụng để nâng cao năng lực xử lý của các ứng dụng nhờ vào cơ chế song song.

- Một luồng là một đơn vị cơ bản của việc sử dụng CPU.

- Nó hình thành gồm: một định danh luồng (thread ID), một bộ đếm chương trình, tập thanh ghi và ngăn xếp.

- Nó chia sẻ với các luồng khác thuộc cùng một quá trình một không gian địa chỉ. Nhờ đó các luồng có thể sử dụng các biến toàn cục, chia sẻ các tài nguyên.

- Cách thức các luồng chia sẻ CPU cũng giống như cách thức của các quá trình.

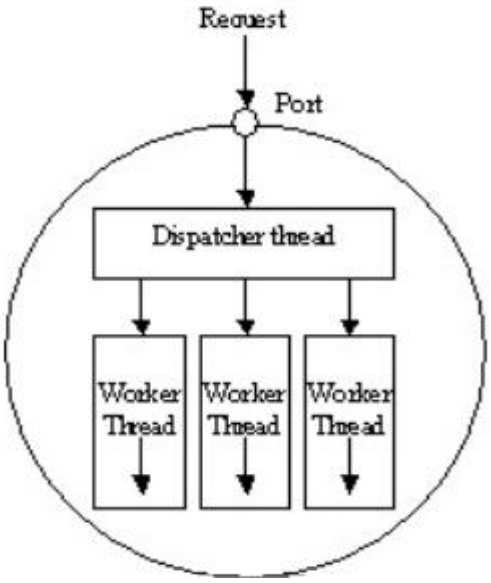
- Một luồng cũng có những trạng thái: đang chạy (`running`), sẵn sàng (`ready`), nghẽn (`blocked`) và kết thúc (`dead`). Một luồng thì được xem như là một quá trình nhẹ.

Trong chương trước, chúng ta đã được tìm hiểu các bước tổng quát của một server phục vụ tuần tự, đến phần này, chúng ta sẽ được tìm hiểu về server phục vụ song song.

Nhờ vào luồng, người ta thiết kế các server có thể đáp ứng nhiều yêu cầu một cách đồng thời.

Các bước tổng quát của một server phục vụ song song

Server phục vụ song song gồm hai phần thực hiện song song nhau:



Hình 4.1. Server ở chế độ song song

Trong mô hình này, server có một luồng phân phát (Dispatcher thread) và nhiều luồng thực hiện (Worker Thread). Luồng phân phát tiếp nhận các yêu cầu kết nối từ clients, rồi chuyển chúng đến các luồng thực hiện còn rảnh để xử lý. Những luồng thực hiện hoạt động song song nhau và song song với cả luồng phân phát, nhờ đó server có thể phục vụ nhiều client một cách đồng thời.

- **Phần 1** (Dispatcher thread): Xử lý các yêu cầu kết nối, lặp lại các công việc sau:
  + Lắng nghe yêu cầu kết nối của clients
  + Chấp nhận một yêu cầu kết nối
    - ✔ Tạo kênh giao tiếp ảo mới với clients
    - ✔ Tạo phần 2 để xử lý các thông điệp yêu cầu của clients.

- **Phần 2** (Worker Thread): Xử lý các thông điệp yêu cầu từ clients, lặp lại các công việc sau:
  + Chờ nhận thông điệp yêu cầu của clients.
  + Phân tích và xử lý yêu cầu.
  + Gửi thông điệp trả lời cho clients.

Phần 2 sẽ kết thúc khi kênh ảo bị xóa đi.

Với mỗi client, trên server sẽ có một **Phần 2** để xử lý yêu cầu của clients. Như vậy tại thời điểm bất kỳ luôn tồn tại một **Phần 1** và 0 hoặc nhiều **Phần 2**

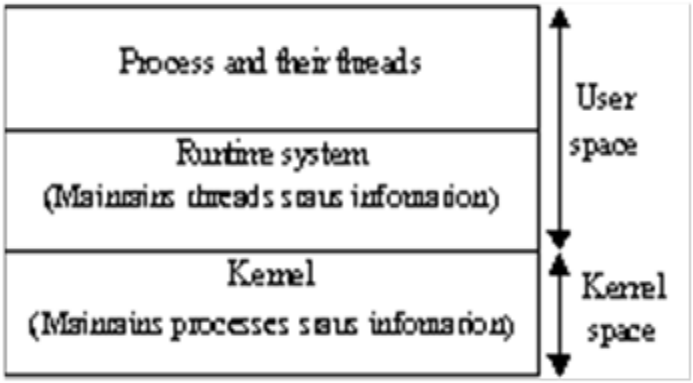
Do *phần 2* thực thi song song với *phần 1* cho nên nó được thiết kế là một thread

- Nhìn từ góc độ hệ điều hành, luồng có thể được cài đặt ở một trong hai mức:

• Trong không gian người dùng (user space)

• Trong không gian nhân (kernel mode)

4.1.1. Tiếp cận luồng ở mức người dùng



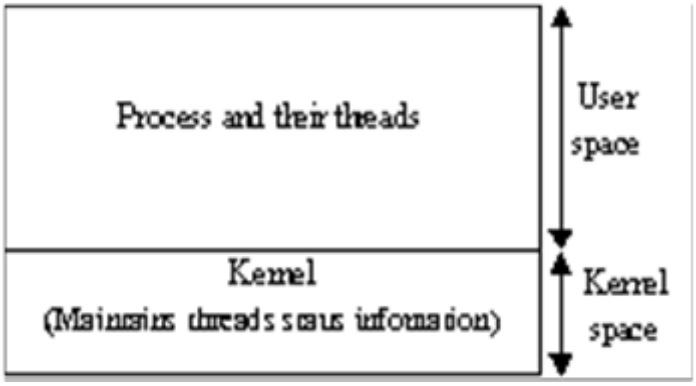
Hình 4.2. Kiến trúc luồng cài đặt ở mức người dùng

Không gian người dùng bao gồm một hệ thống runtime mà nó tập hợp những thủ tục quản lý luồng. Các luồng chạy trong không gian nằm bên trên hệ thống runtime thì được quản lý bởi hệ thống này. Hệ thống runtime cũng lưu giữ một bảng tin trạng thái để theo dõi trạng thái hiện hành của mỗi luồng.

Tương ứng với mỗi luồng sẽ có một mục từ trong bảng, bao gồm các thông tin về trạng thái, giá trị thanh ghi, độ ưu tiên và các thông tin khác về luồng.

Tiếp cận này có hai mức định thời biểu (Scheduling): bộ định thời biểu cho các quá trình nặng và bộ định thời biểu trong hệ thống runtime. Bộ lập biểu của hệ thống runtime chia thời gian sử dụng CPU được cấp cho một quá trình thành những khoảng nhỏ hơn để cấp cho các luồng trong quá trình đó. Như vậy việc kết thúc một luồng thì vượt ra ngoài tầm kiểm soát của kernel hệ thống.

4.1.2. Tiếp cận luồng ở mức hạt nhân hệ điều hành



Hình 4.3. Kiến trúc luồng cài đặt ở mức hệ thống

Trong tiếp cận này không có hệ thống runtime và các luồng thì được quản lý bởi kernel của hệ điều hành. Vì vậy, bảng thông tin trạng thái của tất cả các luồng thì được lưu trữ bởi kernel. Tất cả những lời gọi mà nó làm nghẽn luồng sẽ được bẫy (TRAP) đến kernel. Khi một luồng bị nghẽn, kernel chọn luồng khác cho thực thi. Luồng được chọn có thể cùng một quá trình với luồng bị nghẽn hoặc thuộc một quá trình khác, vì vậy sự tồn tại của một luồng thì được biết bởi kernel và chỉ có một mức lập biểu trong hệ thống.

4.2. Luồng trong Java

Trong Java, luồng là một đối tượng thuộc lớp `java.lang.Thread`. Một chương trình trong java có thể cài đặt luồng bằng cách tạo ra một lớp con của lớp `java.lang.Thread` hoặc cài đặt giao diện `java.lang.Runnable`

4.2.1. Các phương pháp thực hiện luồng

Với Java ta có thể xây dựng các chương trình đa luồng. Một ứng dụng có thể bao gồm nhiều luồng, mỗi luồng được gán một công việc cụ thể, chúng được thực thi đồng thời với các luồng khác.

Có 2 cách để tạo ra luồng :

- *Cách 1* : Thừa kế từ lớp **`java.lang.Thread`**

```
class MyThread extends Thread {
    ....
    public void run() {
        ...
    }
}
```

- *Cách 2* : Cài đặt giao diện **`java.lang.Runnable`**

```
class MyThread implements Runable {
   ...
   public void run() {
        ...
        }
}
```

1. Lớp `Thread`

Lớp `Thread` chứa phương thức khởi tạo `Thread()` cũng như nhiều phương thức hữu ích có chức năng chạy, khởi động, tạm ngưng, tiếp tục, gián đoạn và ngưng luồng. Để tạo ra và chạy một luồng ta cần làm hai bước:

- Mở rộng lớp `Thread` và viết đè phương thức `run()`
- Gọi phương thức `start()` để luồng bắt đầu thực thi

Một số phương thức của `Thread` :

`public void run``()`: được Java gọi để thực thi luồng thi hành, bạn phải viết đè phương thức này để thực thi nhiệm vụ của luồng, bởi vì phương thức *`run`*`()` của lớp `Thread` chỉ là phương thức rỗng.

`public native synchronized void start``()`: khi ta tạo ra luồng nó chưa thực sự chạy cho đến khi phương thức `start()` được gọi, khi `start()` được gọi thì phương thức `run()` cũng được kích hoạt.

`public final void stop``()`: có chức năng ngưng luồng thi hành, phương thức này không an toàn, bạn nên gán `null` vào biến `Thread` để dùng luồng, thay vì sử dụng phương thức `stop()`.

`public final void suspend``()`: có chức năng tạm ngưng luồng, trong Java phương thức này ít được sử dụng, bởi vì phương thức này không nhả tài nguyên mà nó nắm giữ, do vậy có thể nguy cơ dẫn đến deadlock (khoá chết), bạn nên dùng phương thức `wait()` để tạm ngưng luồng thay vì sử dụng phương thức `suspend()`

`public final void resume``()`: tiếp tục vận hành luồng nếu như nó đang bị ngưng, nếu luồng đang thi hành thì phương thức này bị bỏ qua, thông thường phương thức này được dùng kết hợp với phương thức `suspend()`, bạn nên dùng phương thức `notify()` thay vì dùng phương thức `resume()`

public static void sleep(long millis) throws InterruptedException : đặt luồng thi hành vào trạng thái ngủ, trong khoảng thời gian xác định bằng mili giây, chú ý `sleep()` là phương thức tĩnh.

public void interrupt(): làm gián đoạn luồng thi hành

public static boolean isInterrupt(): kiểm tra xem luồng có bị ngắt không

public void setpriority(int p): ấn định độ ưu tiên cho luồng thi hành, độ ưu tiên được xác định là một số nguyên thuộc đoạn [1,10]

public final void wait() throws InterruptException: đặt luồng vào trạng thái chờ một luồng khác, cho đến khi có một luồng khác thông báo thì nó lại tiếp tục, đây là phương thức của lớp cơ sở `Object`

public final void notify`()` : đánh thức luồng đang chờ trên đối tượng này

public final void notifyAll(): đánh thức tất cả các luồng đang chờ trên đối tượng này

isAlive(): trả về `True`, nếu luồng vẫn còn tồn tại (sống)

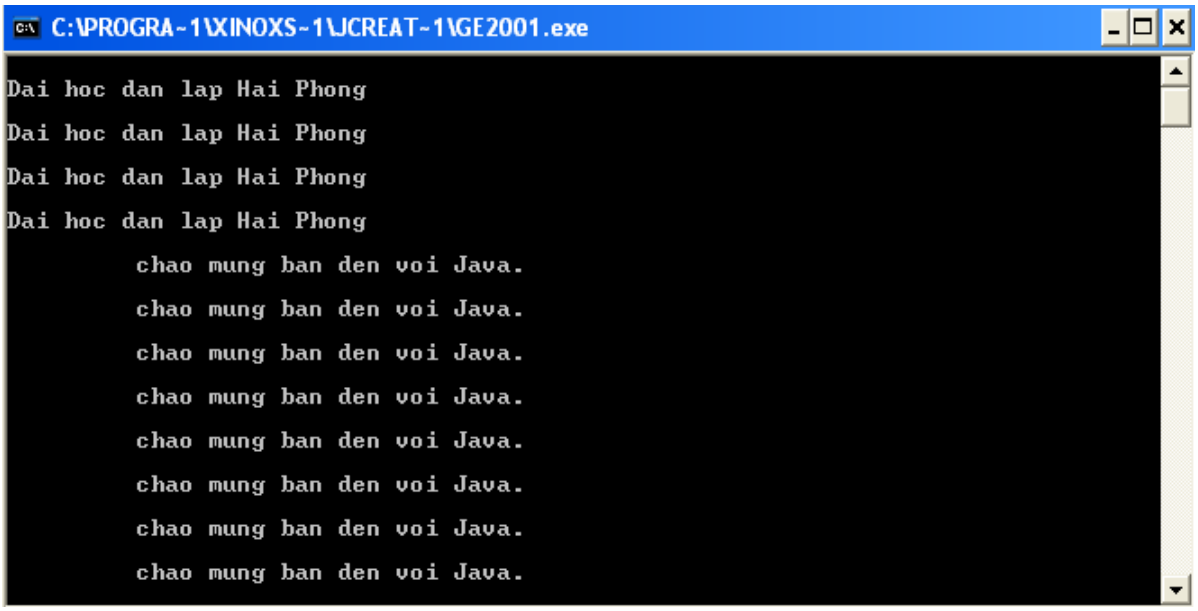
getPriority(): trả về mức ưu tiên của luồng

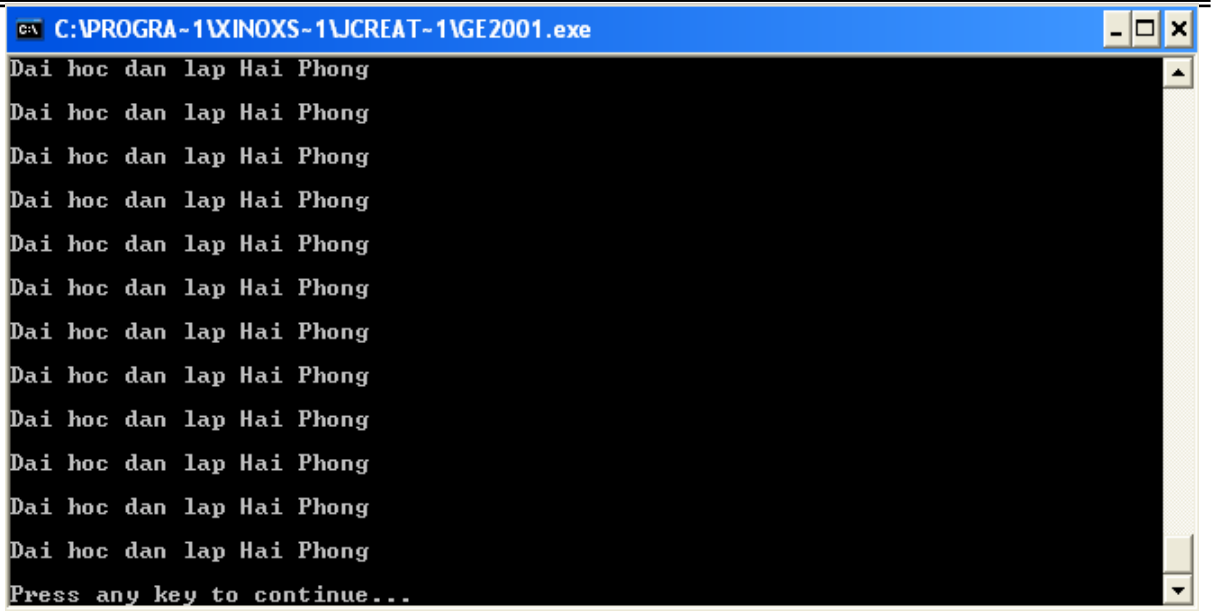
Ví dụ : tạo ra hai luồng thi hành song song, một luồng thực hiện việc in 200 dòng "Dai hoc dan lap Hai Phong"; trong khi luồng này đang thực thi thì có một luồng khác vẫn tiếp tục in 200 dòng chữ "chao mung ban den voi Java"

```
//=========================
import java.net.* ;
import java.io.* ;
public class Hello
{
    public static void main ( String[] args )
    {
        new ChaoDH ().start ();
        new ChaoJV ().start ();
    }
}
```

```
class ChaoDH extends Thread
{
    public void run ()
    {
      for(int i = 1; i <= 200; i++ )
       System.out.println("Dai hoc dan lap Hai Phong \n");
    }
}
class ChaoJV extends Thread
{
    public void run ()
    {
        for ( int i = 1; i <= 200; i++ )
        System.out.println ( "\t chao mung ban den voi Java.\n" );
    }
}
//=========================
```

Khi ta chạy chương trình sẽ thấy các kết quả xen kẽ nhau





2. Giao diện `Runnable`

Do Java không hỗ trợ kế thừa bội, nên nếu chương trình của bạn vừa muốn kế thừa từ một lớp nào đó, lại vừa muốn đa luồng thì bạn bắt buộc phải dùng giao diện `Runnable`, chẳng hạn như bạn viết các applet, bạn vừa muốn nó là applet, lại vừa muốn thực thi nhiều luồng, thì bạn vừa phải kế thừa từ lớp `Applet`, nhưng nếu đã kế thừa từ lớp `Applet` rồi thì bạn không thể kế thừa từ lớp `Thread` nữa.

Ví dụ : ta viết lại ví dụ trên, nhưng lần này ta không kế thừa lớp `Thread` nữa mà triển khai giao diện `Runnable`.

```
import java.net.* ;
import java.io.* ;
public class hello2
{
    public static void main(String[] args)
    {
    Thread t = new Thread (new ChaoDH());
    t.start();
    Thread t1 = new Thread (new ChaoJV());
    t1.start ();
    }
}
```

```
//==================
class ChaoDH implements Runnable
{
  public void run()
  {
    ChaoDH thu = new ChaoDH();
    for ( int i = 1; i <= 200; i++ )
       System.out.println("Dai hoc dan lap Hai Phong\n ");
  }
}
//==================
class ChaoJV implements Runnable
{
    public void run ()
    {
    for ( int i = 1; i <= 200; i++ )
    {
    System.out.println ("\t chao mung ban den voi java. \n" );
    }
    }
}
//=============
```

Kết quả chạy chương trình thu được cũng giống như ví dụ trên.

4.2.2. Độ ưu tiên của các luồng

- Độ ưu tiên của các luồng xác định mức ưu tiên trong việc phân phối CPU giữa các luồng với nhau. Khi có nhiều luồng đang ở trạng thái "`ready`", luồng có độ ưu tiên cao nhất sẽ được thực thi (chuyển sang "`running`").

- Khi một luồng được tạo ra, nó nhận một độ ưu tiên mặc định (bằng 5), đôi khi ta muốn điều chỉnh độ ưu tiên của luồng để đạt được mục đích của ta, thật đơn giản, để đặt độ ưu tiên cho một luồng ta chỉ cần gọi phương thức **`setPriority()`** và

truyền cho nó một số nguyên, số này chính là độ ưu tiên mà bạn cần đặt. Để kiểm tra ta có thể gọi phương thức **`getPriority()`**

- Độ ưu tiên trong Java được định nghĩa bằng các hằng số nguyên theo thứ tự giảm dần như sau:

  + `Thread.MAX_PRIORITY` (giá trị 10)
  + `Thread.NORM_PRIORITY` (giá trị 5)
  + `Thread.MIN_PRIORITY` (giá trị 1)

- Một luồng mới sẽ thừa kế độ ưu tiên từ luồng tạo ra nó.

4.2.3. Nhóm luồng

- Nhóm luồng là một tập hợp gồm nhiều luồng, khi ta tác động đến nhóm luồng (chẳng hạn như tạm ngưng, …) thì tất cả các luồng trong nhóm đều nhận được cùng tác động đó, điều này là tiện lợi khi ta muốn quản lý nhiều luồng thực hiện các tác vụ tương tự nhau.

- Để tạo một nhóm luồng ta cần:

+ Tạo ra một nhóm luồng bằng cách sử dụng phương thức tạo dựng của lớp `ThreadGroup()`

```
ThreadGroup g = new ThreadGroup("ThreadGroupName");
ThreadGroup g = new ThreadGroup(ParentThreadGroup,"ThreadGroupName");
```

Dòng lệnh trên tạo ra một nhóm luồng `g` có tên là "ThreadGroupName", tên của luồng là một chuỗi và không trùng với tên của một nhóm khác.

+ Đưa các luồng vào nhóm luồng dùng phương thức tạo dựng của lớp `Thread()`:

```
Thread =new Thread (g, new ThreadClass(),"ThisThread");
```

4.2.4. Đồng bộ hóa các luồng thi hành

- Tất cả các luồng của một quá trình thì được thực thi song song và độc lập nhau nhưng lại cùng chia sẻ nhau một không gian địa chỉ của quá trình. Chính vì vậy có thể dẫn đến khả năng đụng độ trong việc cập nhật các dữ liệu dùng chung của chương trình (biến, các tập tin được mở).

VD: một luồng có thể cố gắng đọc dữ liệu, trong khi luồng khác cố gắng thay đổi dữ liệu ấy → dữ liệu có thể bị sai.

- Trong những trường hợp này, bạn cần cho phép một luồng hoàn thành trọn vẹn tác vụ của nó, và rồi thì mới cho phép các luồng kế tiếp thực thi. Khi hai hoặc nhiều hơn một luồng cần thâm nhập đến một tài nguyên được chia sẻ, bạn cần chắc chắn rằng tài nguyên đó sẽ được sử dụng chỉ bởi một luồng tại một thời điểm.

- Đồng bộ hoá luồng (`thread synchronization`) giúp cho tại mỗi thời điểm chỉ có một luồng có thể truy nhập vào đối tượng, còn các luồng khác phải đợi .

- Các `Thread` được đồng bộ hoá trong Java sử dụng thông qua một bộ giám sát (monitor). Hãy nghĩ rằng, một monitor là một đối tượng cho phép một `Thread` truy cập vào một tài nguyên, chỉ có một `Thread` sử dụng một monitor tại một thời điểm bất kỳ; các lập trình viên thường nói rằng: `Thread` sở hữu monitor vào thời gian đó.

- Một `Thread` chỉ có thể sở hữu một monitor nếu như không có `Thread` nào đang sở hữu monitor đó. Khi một monitor đang ở trạng thái sẵn sàng thì một `Thread` có thể sở hữu monitor và nó có thể truy cập thẳng đến tài nguyên được tập hợp với monitor đó. Ngược lại, `Thread` sẽ bị tạm treo cho đến khi monitor trở lại trạng thái sẵn sàng. Các lập trình viên nói rằng `Thread` đang chờ monitor.

- Bạn thấy các thao tác với monitor có vẻ rất phức tạp đúng không? nhưng đừng ngại nó vì tất cả các thao tác của việc yêu cầu một monitor được Java tự động giải quyết cho bạn và nó trong suốt với người dùng.

- Có hai cách để đồng bộ hoá các luồng: sử dụng method được đồng bộ hóa hoặc sử dụng phát biểu được đồng bộ hóa.

Sử dụng method được đồng bộ hóa:

Tất cả các đối tượng trong Java đều có một monitor. Một `Thread` có một monitor bất kỳ khi nào một method được bổ sung từ khóa `synchronized` ở đầu method đó được gọi.

Khi một luồng gọi phương thức `synchronized`, đối tượng sẽ bị khoá. Khi luồng đó thực hiện xong phương thức, đối tượng sẽ được mở khoá.

Trong khi thực thi phương thức `synchronized`, một luồng có thể gọi **`wait()`** để chuyển sang trạng thái chờ cho đến khi một điều kiện nào đó xảy ra. Khi luồng đang chờ, đối tượng sẽ không bị khoá.

Khi thực hiện xong công việc trên đối tượng, một luồng cũng có thể thông báo (**notify**) cho các luồng khác đang chờ để truy nhập đối tượng.

Sử dụng phát biểu được đồng bộ hóa:

Việc đồng bộ hoá một phương thức là cách tốt nhất để hạn chế việc sử dụng một phương thức tại một thời điểm. Tuy nhiên sẽ có những trường hợp mà bạn không thể đồng bộ hoá một phương thức, chẳng hạn như khi bạn sử dụng một `class` được cung cấp bởi bên thứ ba. Trong những trường hợp như thế, bạn không được phép truy nhập vào định nghĩa lớp, sẽ ngăn bạn sử dụng từ khoá `synchronized`.

Một cách khác để sử dụng từ khóa `synchronized` là sử dụng phát biểu được đồng bộ hóa. Một phát biểu được đồng bộ hóa chứa một `block` được đồng bộ hóa, mà bên trong đó đặt những đối tượng và những phương thức được đồng bộ hóa. Gọi các method chứa block được đồng bộ hóa xảy ra khi một `Thread` có được monitor của đối tượng.

CHƯƠNG 5 : CHƯƠNG TRÌNH ỨNG DỤNG

Trong chương này sẽ trình bầy một ứng dụng của cơ chế thực hiện đa luồng trong ngôn ngữ lập trình Java, đó là xây dựng chương trình truy nhập cơ sở dữ liệu Web.

5.1. Giới thiệu

Truy nhập cơ sở dữ liệu Web là một nhu cầu phổ biến trong thực tế. Các cơ sở dữ liệu Web thường được đặt trên các máy chủ có cấu hình mạnh, trong khi đó việc truy nhập vào cơ sở dữ liệu để lấy dữ liệu thường được thực hiện trên các máy trạm trên mạng thông qua trình duyệt. Vậy làm thế nào để thực hiện đáp ứng được nhu cầu đa truy nhập vào cơ sở dữ liệu của phía clients lại vừa có thể đảm bảo tính bảo mật cho cơ sở dữ liệu trên máy chủ?

Trong Java hỗ trợ phương pháp lập trình đa luồng và lập trình socket (TCP) là một phương pháp cho phép chúng ta có thể đa truy nhập cơ sở dữ liệu Web. Phương pháp này dựa trên nguyên lý hoạt động của `Thread`, socket TCP; socket được sử dụng để kết nối giữa clients với một thành phần trung gian (middle ware).

Middle ware trong chương trình ứng dụng này là một chương trình Java ở dạng độc lập (Java Application) chạy trên phía server. Nó có nhiệm vụ lắng nghe yêu cầu kết nối từ phía clients và thực hiện các yêu cầu truy nhập cơ sở dữ liệu của clients, sau đó sẽ trả lại kết quả cho clients.

Sở dĩ phải sử dụng middle ware là do trong Java không cho phép các chương trình dạng Applet được truy nhập vào tài nguyên cục bộ, chỉ có các chương trình dạng độc lập (Application) mới có quyền truy nhập. Điều này đảm bảo cho tính an toàn của dữ liệu, tránh cho việc người sử dụng làm sai lệch thông tin.

Ưu điểm của phương pháp này là chúng ta chỉ mất thời gian mở kết nối tới server trong lần truy nhập đầu tiên, các lần tiếp theo chúng ta không cần phải mở kết nối lại mà chỉ gửi các yêu cầu truy vấn tới server và chờ kết quả trả về thông qua middle ware.

5.2. Mô hình chung truy nhập cơ sở dữ liệu Web

Hình 5.1. Mô hình chung truy nhập cơ sở dữ liệu Web

Hình 5.1 là mô hình tổng quát của một ứng dụng truy nhập cơ sở dữ liệu Web. Trong mô hình này gồm ba lớp:

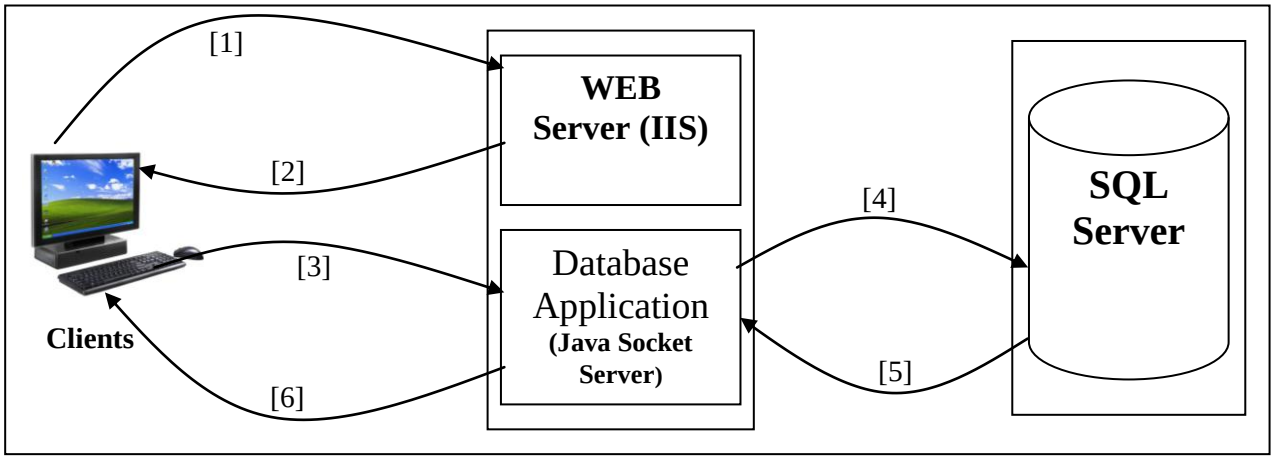
- Lớp thứ nhất (clients): Khi có yêu cầu truy nhập cơ sở dữ liệu Web, phía clients sử dụng trình duyệt Web để kết nối đến Web Server và gửi yêu cầu lấy dữ liệu.
- Lớp thứ hai (middle ware): bao gồm Web Server và một ứng dụng cơ sở dữ liệu. Khi nhận được yêu cầu của phía clients, ứng dụng cơ sở dữ liệu sẽ kết nối đến hệ quản trị cơ sở dữ liệu để lấy dữ liệu theo yêu cầu của clients và trả lại cho phía clients.
- Lớp thứ ba (Database Server): là một hệ quản trị cơ sở dữ liệu, hỗ trợ các câu truy vấn dữ liệu (dạng SQL), lưu trữ dữ liệu cần phục vụ.

Web Server và ứng dụng cơ sở dữ liệu cùng với Database Server có thể đặt trên cùng một máy hoặc trên các máy khác nhau.

5.3. Chương trình ứng dụng

5.3.1. Mô hình và cơ chế hoạt động

1. Mô hình



Hình 5.2. Mô hình chương trình truy nhập cơ sở dữ liệu Web

2. Cơ chế hoạt động

Cơ chế hoạt động kết nối và truy vấn dữ liệu của chương trình được thực hiện bao gồm 6 bước:

[1]. Phía clients sử dụng trình duyệt Web kết nối đến Web Server.

[2]. Phía Web Server trả lại clients trang Web được phía clients yêu cầu có nhúng applet (mô đun clients) ở bên trong.

[3]. Applet tải từ phía Web Server về được trình duyệt Web phía clients kích hoạt, tạo kết nối (socket) đến Socket Server và gửi thông tin (mã sinh viên) đến Socket Server.

[4]. Java Socket Server thực hiện truy vấn cơ sở dữ liệu đến SQL Server bằng các câu lệnh truy vấn thông thường để lấy dữ liệu theo yêu cầu của phía clients.

[5]. Database Server (SQL server) trả dữ liệu kết quả theo yêu cầu cho Socket Server.

[6]. Java Socket Server trả dữ liệu kết quả về cho phía clients trên kết nối mới được tạo ra.

5.3.2. Thiết kế và cài đặt cơ sở dữ liệu thử nghiệm

Xây dựng một cơ sở dữ liệu trên hệ quản trị cơ sở dữ liệu SQL Server 2000 có tên là " QLSV_H ".

* Cơ sở dữ liệu bao gồm 3 bảng được thiết kế như sau:

- **SINH VIÊN** (mã sinh viên, họ tên, ngày sinh, giới tính, địa chỉ, lớp, tên ngành)
- **MÔN HỌC** (mã môn, tên môn, đơn vị học trình, tên ngành)
- **ĐIỂM THI** (mã sinh viên, mã môn, điểm thi, lần thi, ghi chú)

* Cài đặt cơ sở dữ liệu

tbl_sinhvien

	Column Name	Data Type	Length	Allow Nulls
🔑	masv	char	10	
	hoten	nvarchar	50	
	ngaysinh	datetime	8	
	gioitinh	nvarchar	5	
	diachi	nvarchar	50	✓
	lop	char	10	✓
	tennganh	nvarchar	30	✓

SQL Server Enterprise Manager - [Data in Table 'tbl_sinhvien' in 'QLSV_H' on '(local)']

File Window Help

masv	hoten	ngaysinh	gioitinh	diachi	lop	tennganh
080080	Hong Thu	3/3/1985	nam	hhhh	CB801	cb
080081	Tuoi	6/6/1985	nu	Ninh Giang, Hai Duong	CB801	cb
080082	Thanh Tung	11/11/1984	nam	Lach Tray, Hai Phong	CT901	cntt
080090	Thanh Van	11/20/1986	nu	Kien Thuy, Hai Phong	CB802	cb
090009	Hoang Yen	9/9/1987	nu	Hai Hau, Nam Dinh	VH901	vh
090079	Viet Cong	1/1/1987	nam	Ngo Quyen, Hai Phong	CT902	cntt
090093	Van Hao	2/2/1986	nam	Le Chan, Hai Phong	CT902	cntt
090098	Hoàng	10/9/1987	nu	An Duong, Hai Phong	CT902	cntt
090103	Sơn Lâm	5/5/1987	nam	Hong Bang, Hai Phong	VH902	vh
090115	Phuong	4/4/1986	nu	Chi Linh, Hai Duong	VH901	vh

tbl_monhoc

Column Name	Data Type	Length	Allow Nulls
mamon	char	10	
tenmon	nvarchar	50	
dvht	int	4	
tennganh	nvarchar	30	

SQL Server Enterprise Manager - [Data in Table 'tbl_monhoc' in 'QLSV_H' on '(local)']

File Window Help

mamon	tenmon	dvht	tennganh
mh01	asp	5	cntt
mh02	anh CN	5	cntt
mh03	LT C	5	cntt
mh04	LT java	4	cntt
mh05	tieng viet	5	vh
mh06	phat am	3	vh
mh07	hoa hoc	3	cb
mh08	bao quan	5	cb
mh09	oracle	3	cntt
mh10	toan A2	5	cntt

tbl_diemthi

Column Name	Data Type	Length	Allow Nulls
masv	char	10	
mamon	char	10	
diemthi	float	8	
lanthi	smallint	2	✓
ghichu	nvarchar	20	✓

SQL Server Enterprise Manager - [Data in Table 'tbl_diemthi' in 'QLSV_H' on '(local)']

File Window Help

masv	mamon	diemthi	lanthi	ghichu
090098	mh01	8	1	<NULL>
090098	mh02	8	1	<NULL>
080080	mh07	6	1	<NULL>
080080	mh08	8	1	<NULL>
090115	mh05	9	1	<NULL>
090103	mh05	8	1	<NULL>
090093	mh01	8	1	<NULL>
090079	mh02	9	1	<NULL>
080082	mh03	7	1	<NULL>
080082	mh04	8	2	thi lai
090115	mh06	7	1	<NULL>
090098	mh09	7	1	<NULL>
090098	mh10	5	1	<NULL>
090093	mh04	8	1	<NULL>
090079	mh09	9	1	<NULL>

5.3.3. Thiết kế chương trình

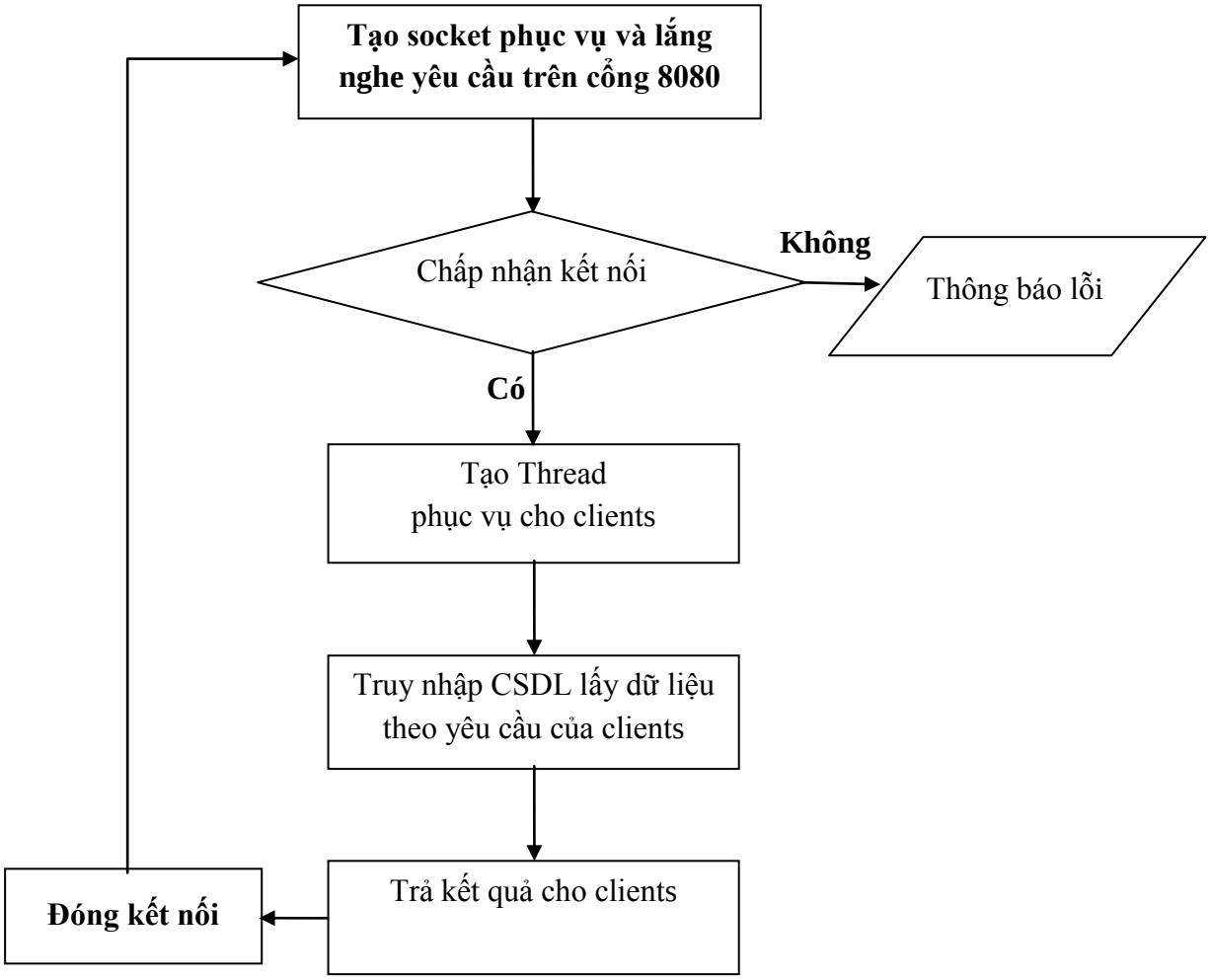
Chương trình ứng dụng được viết bằng ngôn ngữ lập trình Java

1. Phía server

Mô đun server là Application, khi được kích hoạt nó sẽ tạo ra một server socket trên một cổng xác định và lắng nghe các yêu cầu kết nối từ phía clients.

Khi có yêu cầu kết nối từ phía clients, nếu server socket không chấp nhận kết nối thì thông báo lỗi lên màn hình; nếu chấp nhận kết nối thì server socket sẽ tạo ra một `Thread` đáp ứng yêu cầu của clients

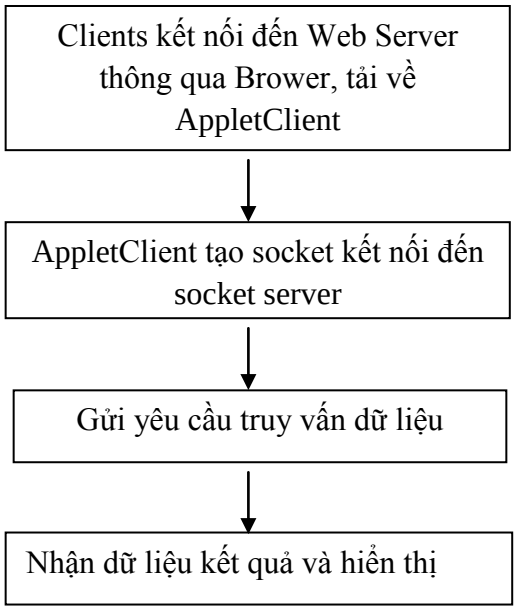
Nó tạo ra một client socket mới để trả lời cho clients. Client socket của server sẽ kết nối đến cơ sở dữ liệu SQL Server, thực hiện truy vấn để lấy dữ liệu theo yêu cầu của phía clients và trả lại dữ liệu kết quả cho phía clients thông qua client socket.



Hình 5.3. Sơ đồ thiết kế của mô đun phía server

2. Phía clients

Mô đun clients là một Applet được đặt trong cùng một thư mục với mô đun phía server và nó được đưa lên Web Server (IIS). Phía clients kết nối với Web Server thông qua trình duyệt Web và tải Applet này về trình duyệt Web. Tại trình duyệt Web của clients, Applet được kích hoạt và tạo ra một socket kết nối tới server socket ở phía server thông qua địa chỉ của máy tính chạy socket server và số hiệu cổng của socket server (đã biết trước). Khi kết nối được chấp nhận, Applet gửi yêu cầu truy vấn dữ liệu cho server socket, sau đó nhận dữ liệu kết quả và hiển thị.



Hình 5.4. Sơ đồ thiết kế của mô đun phía Client

5.3.4. Một số giao diện chính

Dưới đây là kết quả của chương trình khi chạy trên mạng máy tính:

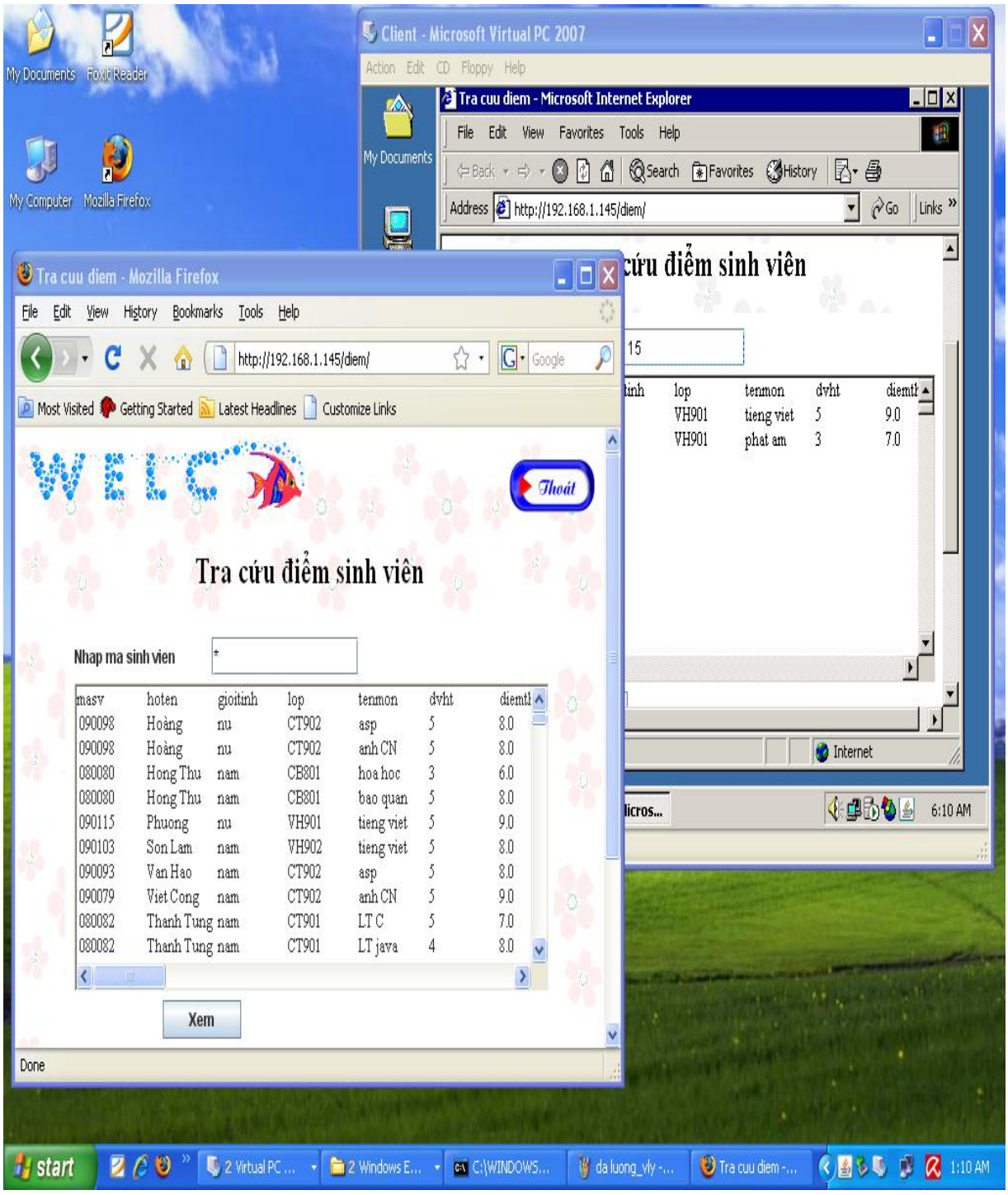
- Trường hợp 1: Mạng máy tính bao gồm một máy tính vật lý và hai máy ảo
- Trường hợp 2: Mạng máy tính chỉ gồm các máy tính vật lý.

1. Kết quả của chương trình khi chạy trên một máy tính vật lý và hai máy ảo

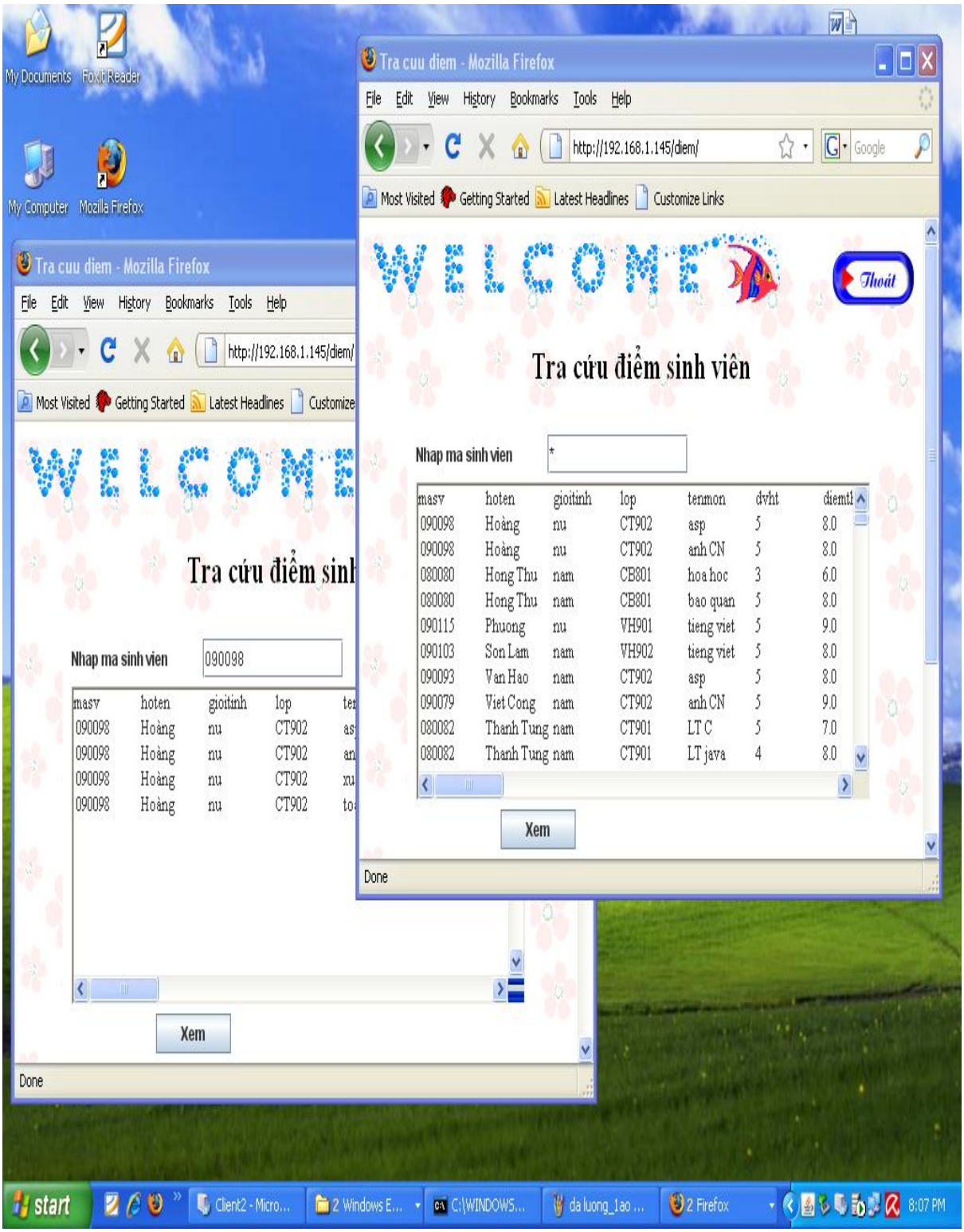
- Máy ảo *Client2* đóng vai trò là lớp thứ nhất (máy trạm)

- Máy vật lý đóng vai trò là lớp thứ hai (middle ware), bao gồm Web Server (IIS) và Database Application (Java Socket Server)

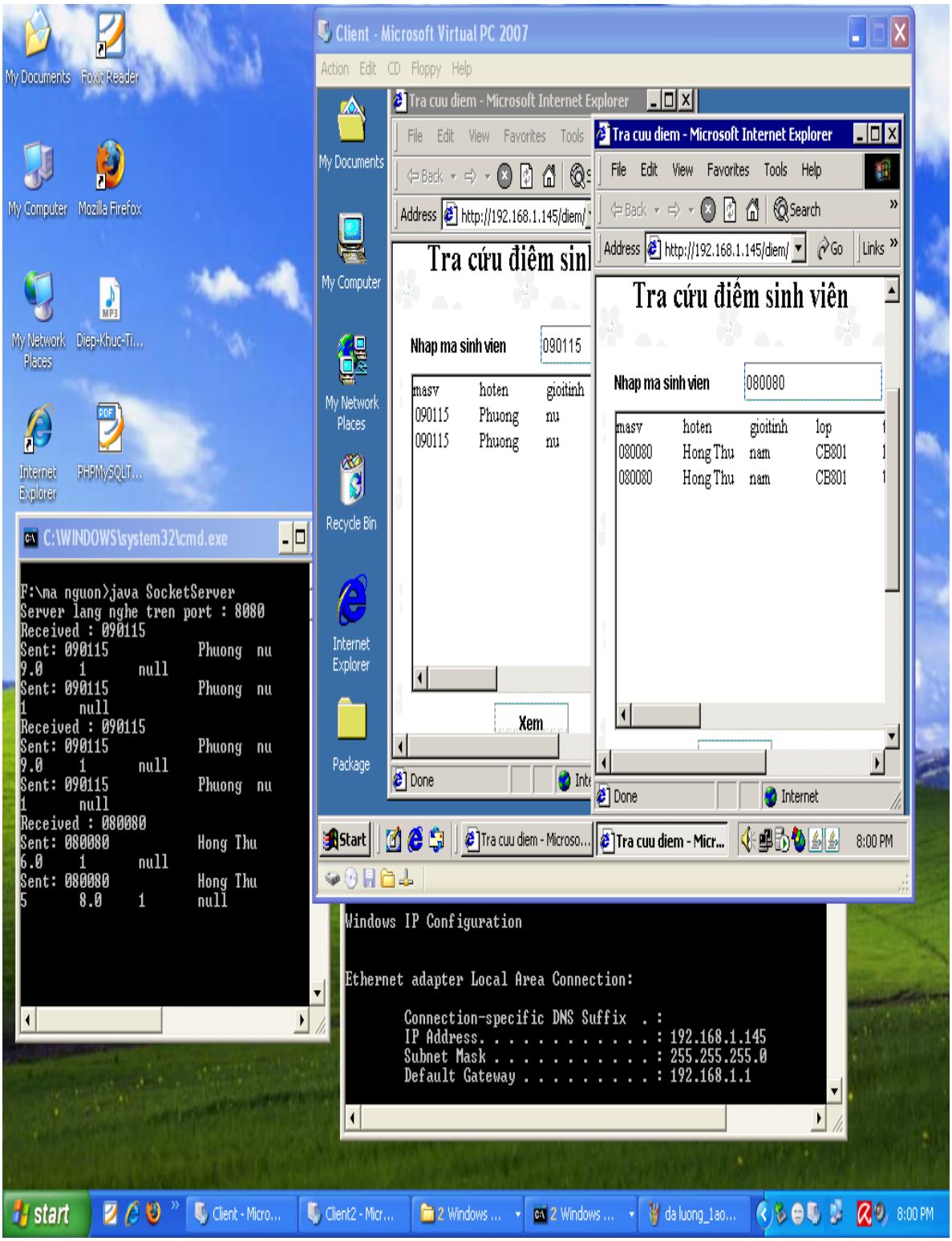
- Máy ảo *Client* là lớp thứ ba (Database Server) được cài đặt hệ quản trị cơ sở dữ liệu SQL Server 2000



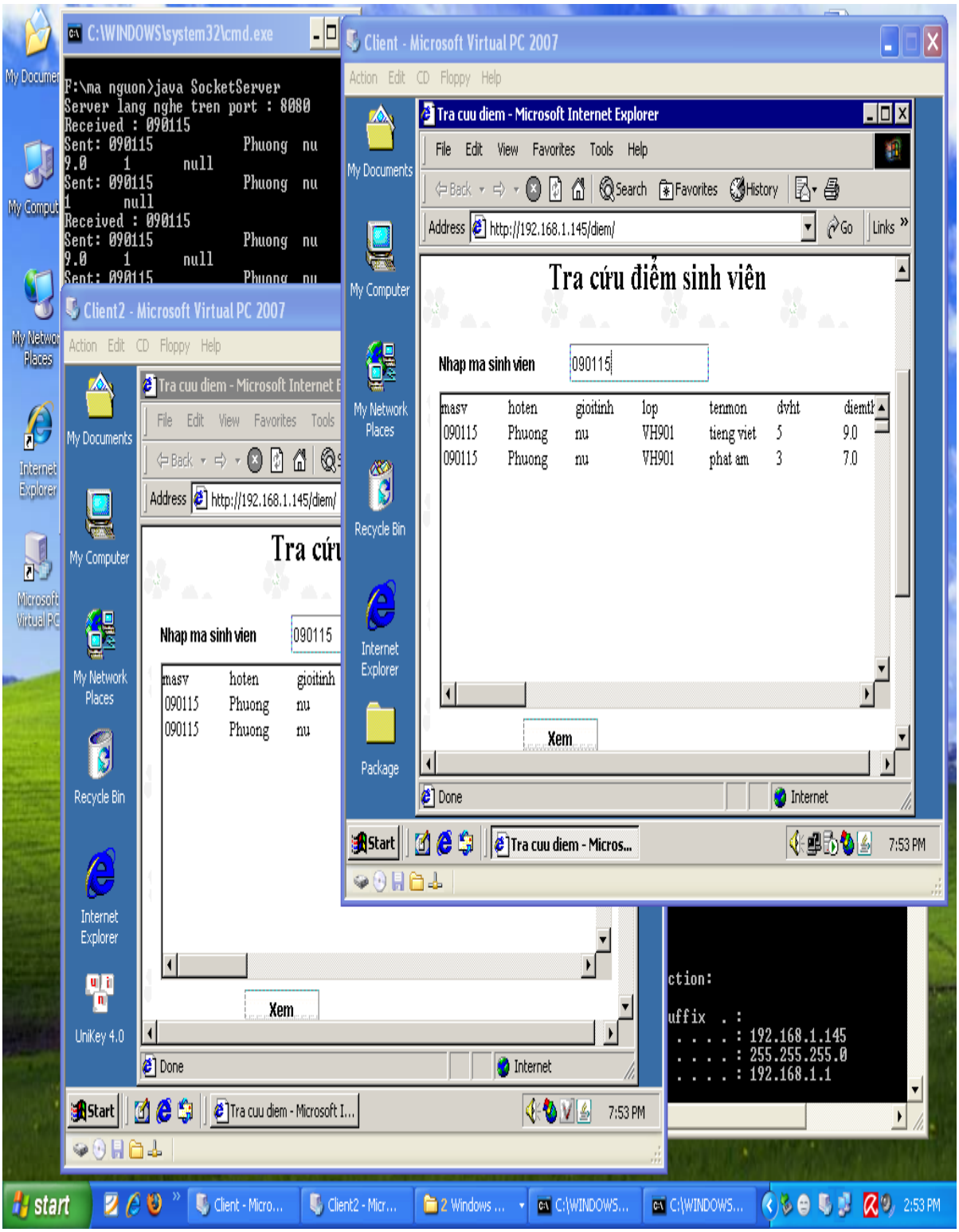
Hình 5.5. Máy vật lý và máy ảo Client cùng truy nhập cơ sở dữ liệu



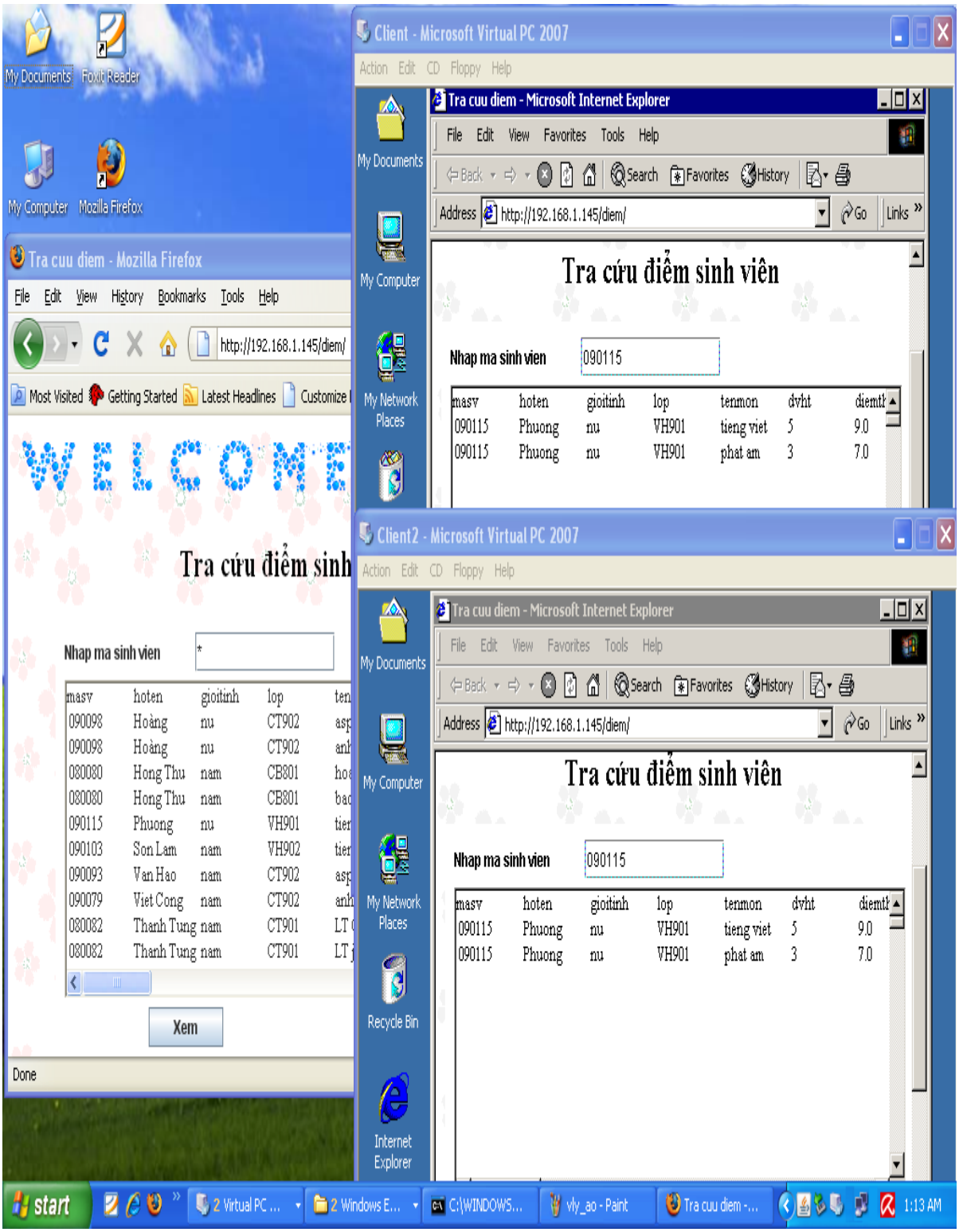
Hình 5.6. Máy vật lý truy nhập cơ sở dữ liệu



Hình 5.7. Máy ảo Client truy nhập cơ sở dữ liệu



Hình 5.8. Hai máy ảo cùng truy nhập cơ sở dữ liệu



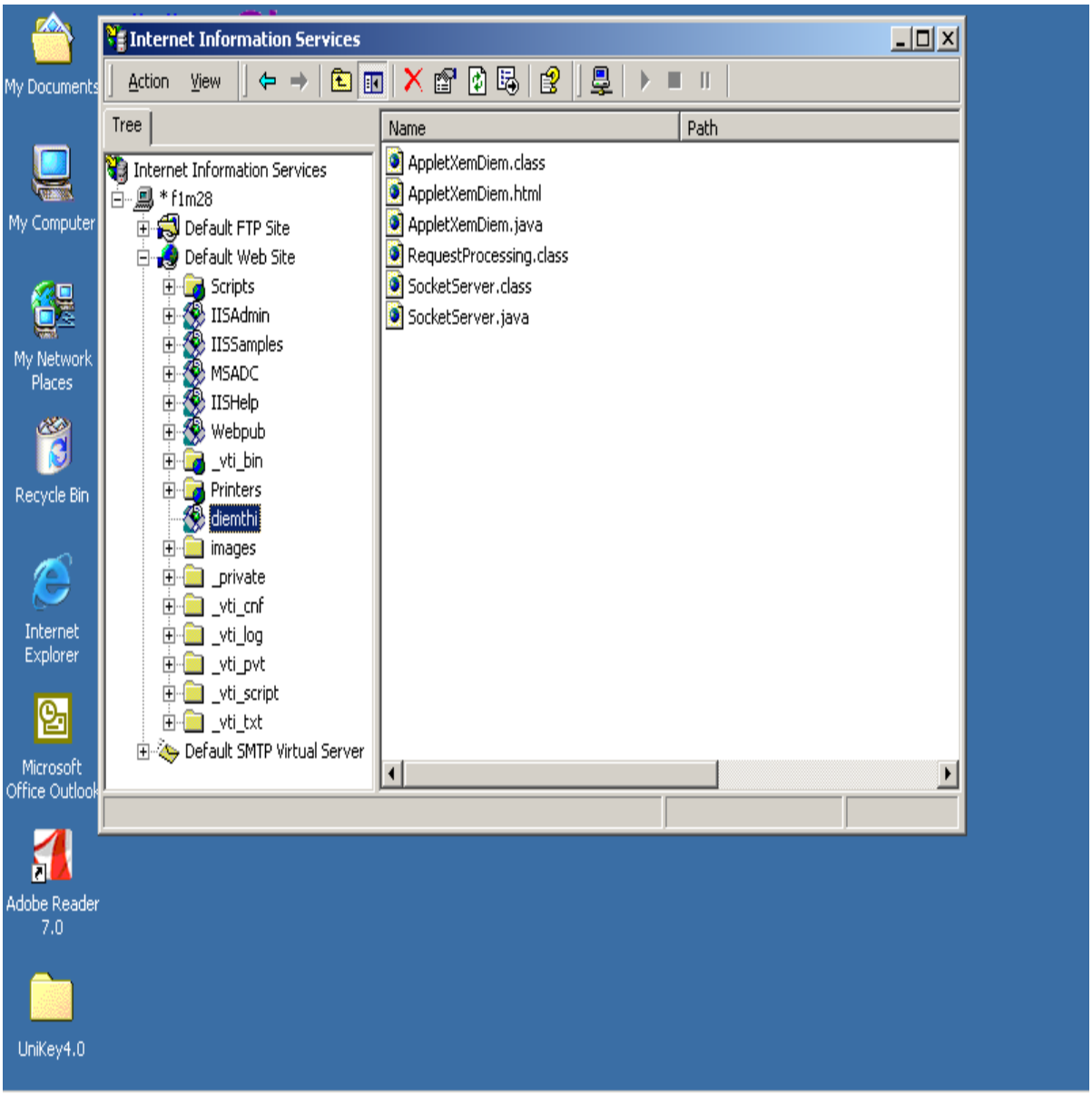
Hình 5.9. Máy vật lý và hai máy ảo cùng truy nhập cơ sở dữ liệu

2. Kết quả của chương trình khi chạy trên các máy tính vật lý

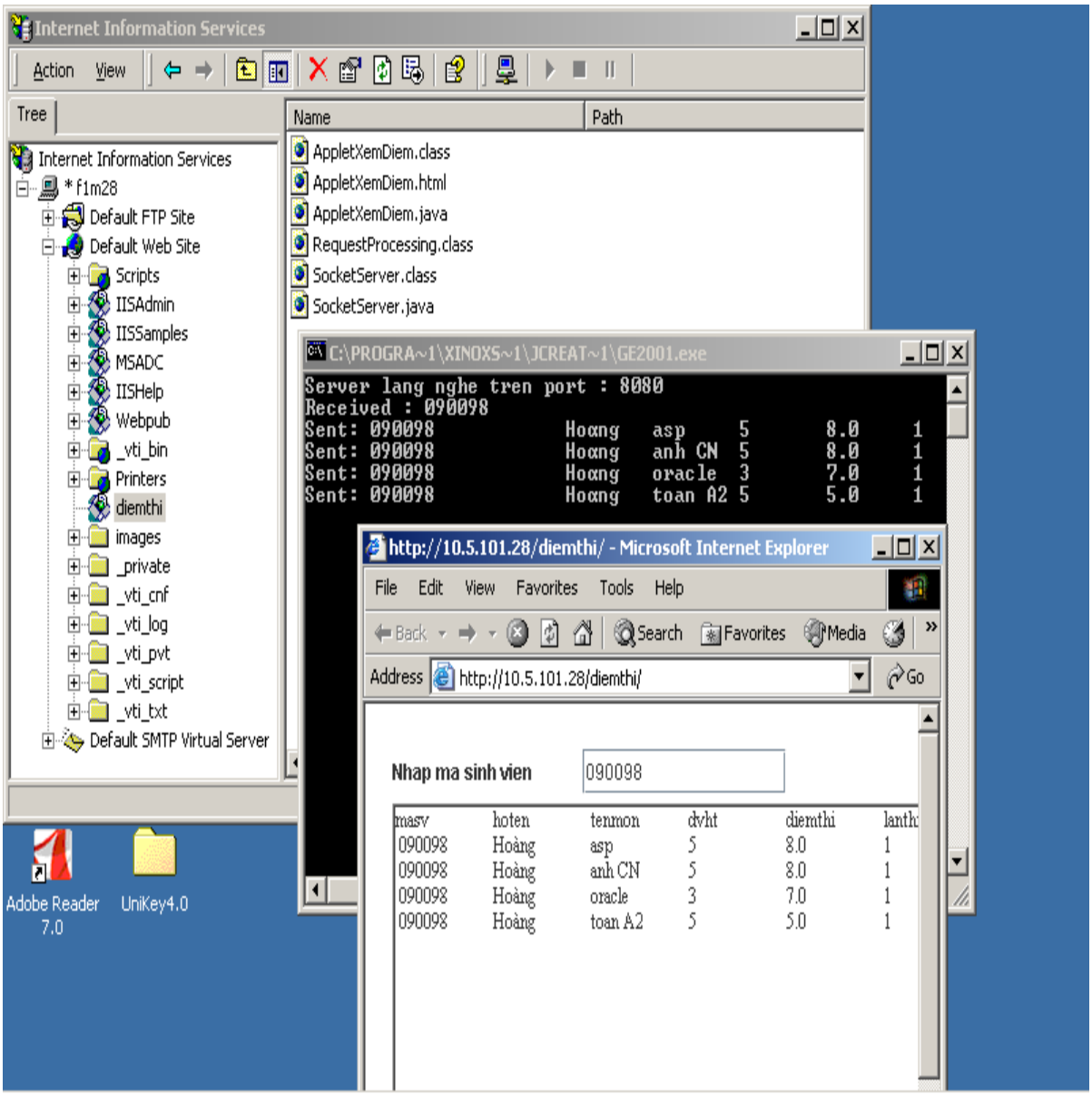
- Máy tính có địa chỉ *10.5.101.29* đóng vai trò là lớp thứ nhất (Clients)

- Máy tính có địa chỉ *10.5.101.28* đóng vai trò là lớp thứ hai (middle ware), bao gồm Web Server (IIS) và Database Application (Java Socket Server)

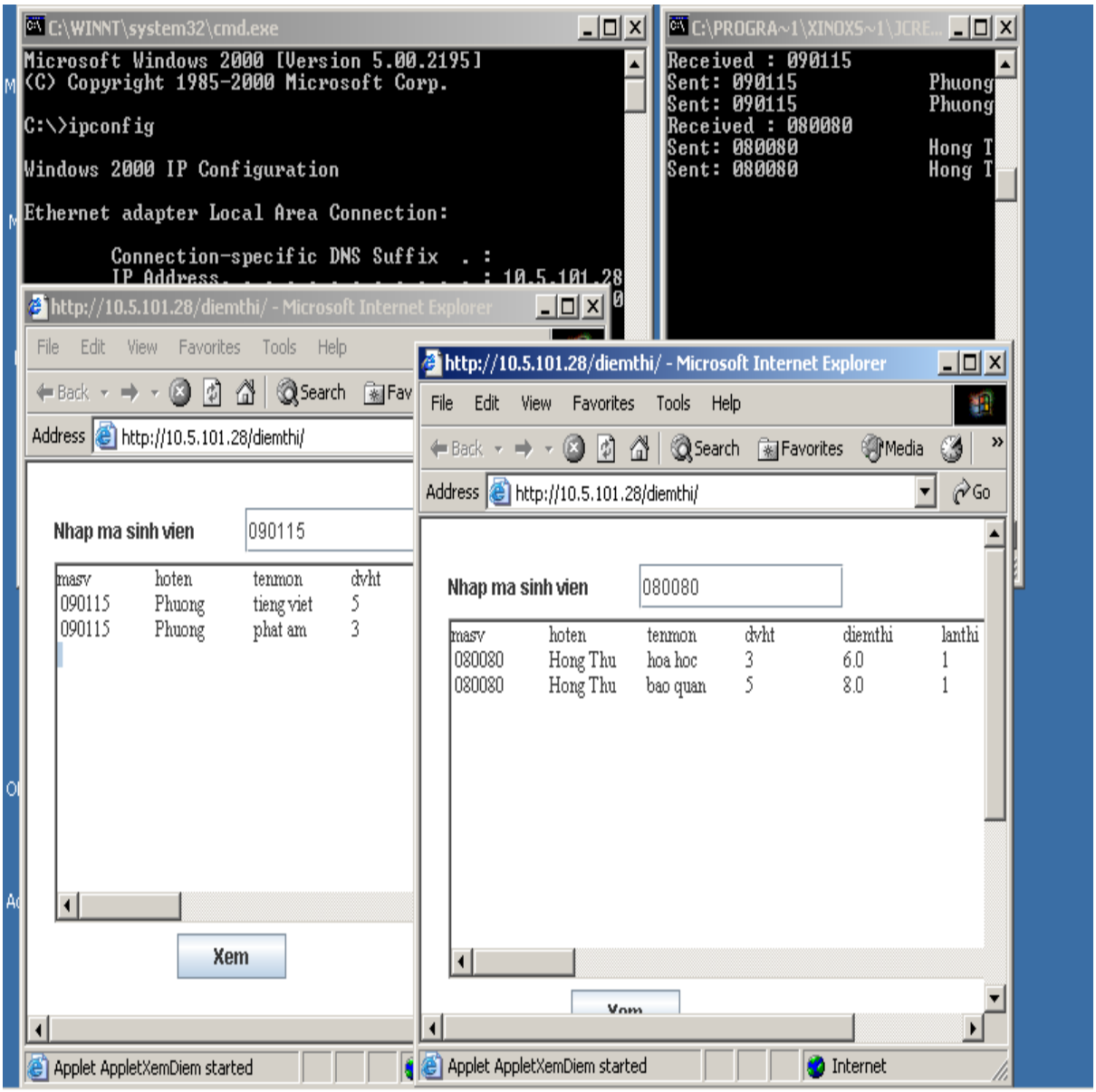
- Máy tính có địa chỉ *10.5.101.30* là lớp thứ ba (Database Server) được cài hệ quản trị cơ sở dữ liệu SQL Server 2000



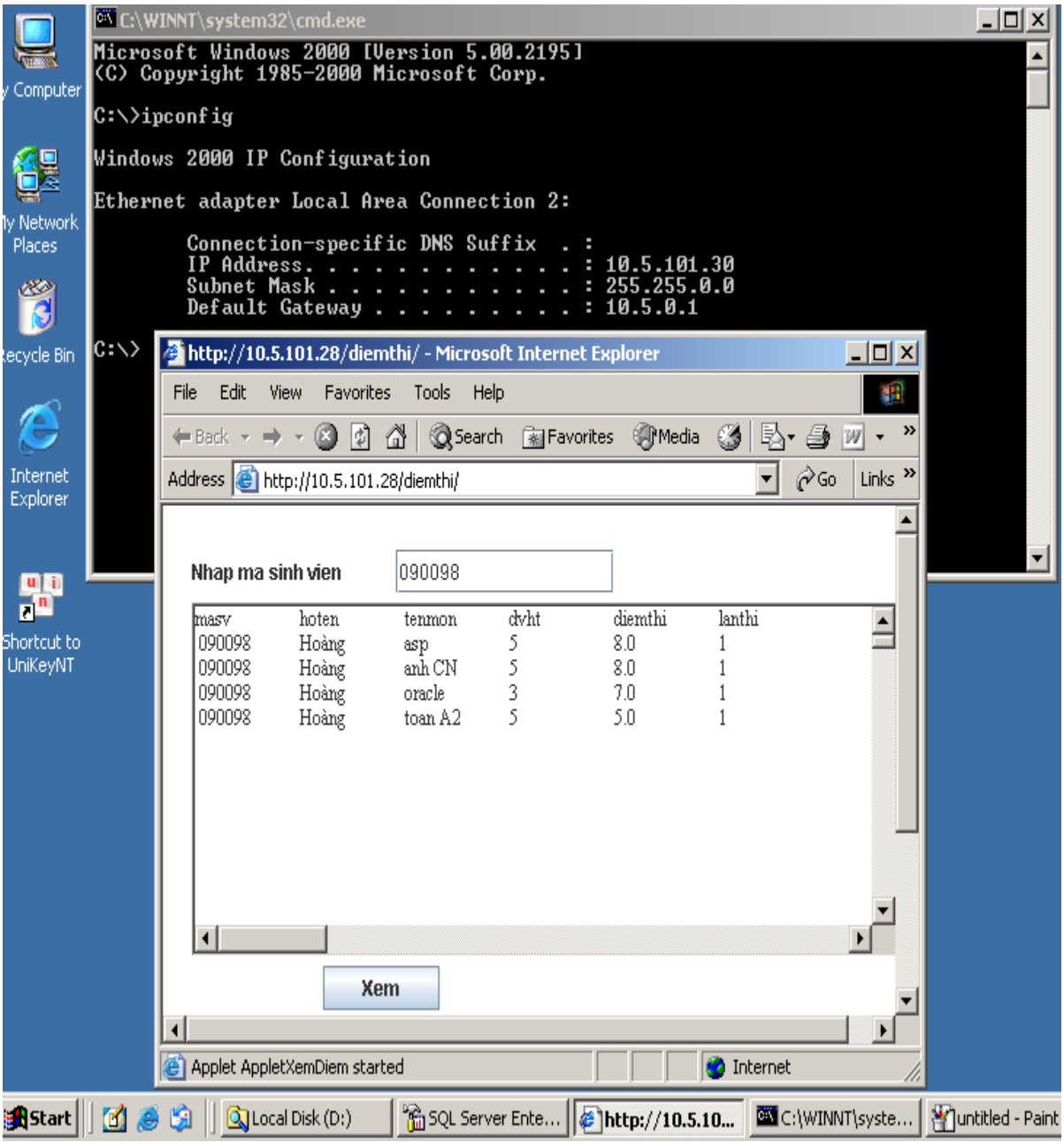
Hình 5.10. Web Server và Database Application đặt trên máy 10.5.101.28



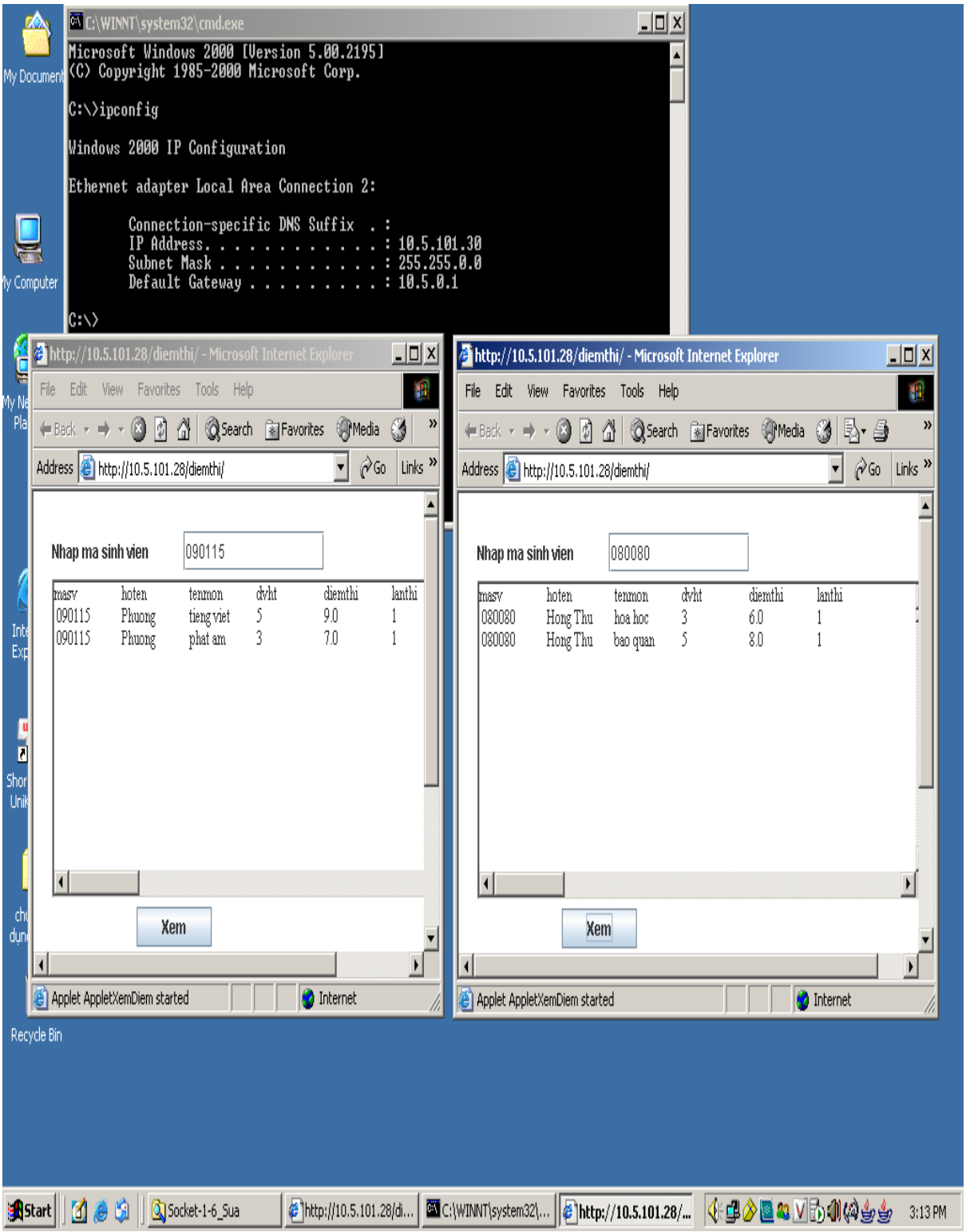
Hình 5.11.(a) Máy 10.5.101.28 khởi động Socket Server và truy nhập cơ sở dữ liệu



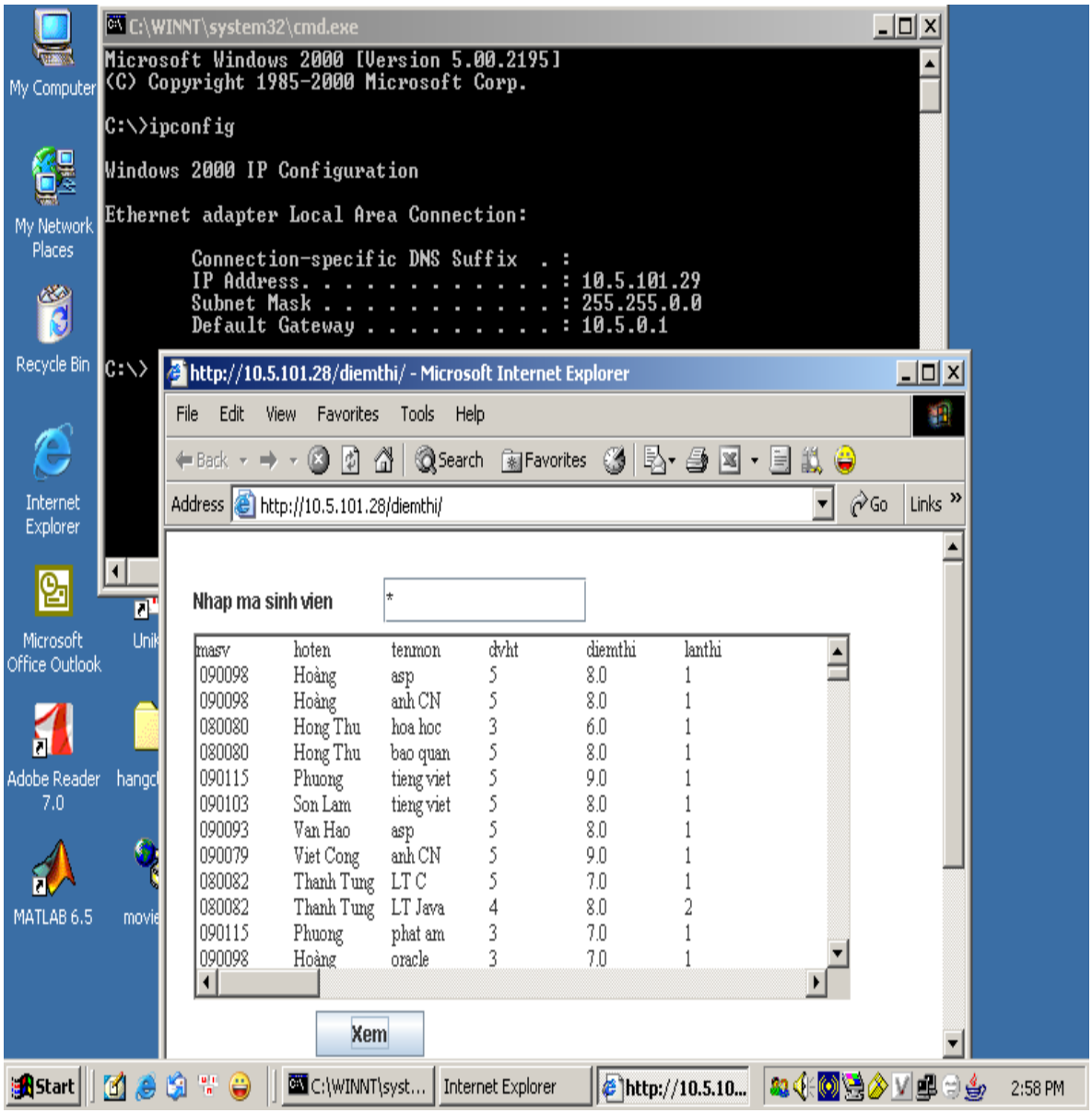
Hình 5.11.(b) Máy 10.5.101.28 khởi động Socket Server và truy nhập cơ sở dữ liệu



Hình 5.12.(a) Máy 10.5.101.30 truy nhập cơ sở dữ liệu bằng trình duyệt



Hình 5.12.(b) Máy 10.5.101.30 truy nhập cơ sở dữ liệu bằng trình duyệt



Hình 5.13. Máy 10.5.101.29 truy nhập cơ sở dữ liệu bằng trình duyệt

5.4. Nhận xét

Chương trình ứng dụng truy nhập cơ sở dữ liệu Web trên có một số ưu điểm sau:

** Phía clients:*

- Không cần cài đặt thêm bất kỳ một mô đun phần mềm nào, chỉ cần có trình duyệt Web là đủ. Do đó chương trình dễ dàng sử dụng với người dùng mà không cần đòi hỏi trình độ cao về công nghệ thông tin.

- Vì mô đun chương trình phía clients là một Applet nên nó không được phép truy nhập vào các tài nguyên cục bộ của máy clients → an toàn cho máy khách.

** Phía server:*

- Toàn bộ các mô đun chương trình Web Server, Socket Server, AppletClient được đặt trong cùng một thư mục và đặt trên cùng một máy, do đó thuận lợi cho công tác cài đặt, nâng cấp, bảo trì chương trình.

- Phía clients muốn lấy được dữ liệu từ cơ sở dữ liệu thì phải truy nhập thông qua thành phần trung gian là Java Socket Server, do đó cơ sở dữ liệu phía server được bảo mật.

* Do chương trình ứng dụng được viết dựa trên nền Web nên có thể được triển khai và sử dụng trên liên mạng (Internet)

KẾT LUẬN

Lập trình đa luồng là một phương pháp tốt để xây dựng các chương trình xử lý song song chạy trên một máy tính chip đơn. Đề tài "*Tìm hiểu lập trình đa luồng trong Java và ứng dụng*" đã đạt được những thành công nhất định. Về cơ sở lý thuyết, đồ án đã trình bầy được các nội dung về mạng máy tính, sơ lược về ngôn ngữ Java, lập trình Socket TCP nói chung và lập trình Socket TCP trong Java nói riêng; các nội dung liên quan đến luồng và lập trình đa luồng trong Java. Về ứng dụng đồ án đã giới thiệu, đưa ra được mô hình chương trình, cơ chế hoạt động và cài đặt thành công chương trình truy nhập cơ sở dữ liệu Web.

Bên cạnh đó đồ án cũng phân tích chi tiết cách thiết kế, cài đặt chương trình cho các độc giả quan tâm có thể tiến hành làm thực nghiệm dễ dàng. Java là một ngôn ngữ mạnh mẽ, tính bảo mật cao và độc lập với nền, do đó chương trình ứng dụng của đồ án có thể dễ dàng chạy trên các hệ thống khác nhau mà không phải lập trình lại. Tuy nhiên, với thời gian và trình độ còn nhiều hạn chế nên đồ án vẫn còn một số vấn đề chưa kịp giải quyết như chưa hiển thị được font tiếng Việt trên form, chưa có ví dụ minh họa cho lý thuyết nhóm luồng và đồng bộ hóa giữa các luồng, cơ sở dữ liệu chưa đủ lớn.

Trong tương lai em sẽ tiếp tục tìm hiểu, khắc phục các hạn chế, mở rộng và hoàn thiện chương trình.

TÀI LIỆU THAM KHẢO

Tài liệu tiếng Việt

[1]. *Lập trình hướng đối tượng với Java*

TS. Đoàn Văn Ban - Viện Công nghệ thông tin

[2]. *Giáo trình lập trình truyền thông*

Biên soạn: Ngô Bá Hùng, Nguyễn Công Huy - Đại học Cần Thơ

[3]. *Mạng thông tin máy tính- Kiến trúc, nguyên tắc và hiệu suất hoạt động*

Vũ Duy Lợi - Nhà xuất bản Đại học Quốc gia Hà Nội

[4]. *Đề cương bài giảng Java cơ sở*

Đại học sư phạm kỹ thuật Hưng Yên

[5]. *JAVA lập trình mạng*

Nguyễn Phương Lan và Hoàng Đức Hải - NXB Giáo Dục

[6]. *Học nhanh kỹ thuật lập trình Java*

Nguyễn Viết Linh, Đậu Quang Tuấn - Xí nghiệp in Bến Tre

Tài liệu tiếng Anh

[1]. *Java Network Programming*

Elliotte Rusty Harold

[2]. *Programming the Internet with Java*

Darrel Ince & Adam Freemat, Addison-Wesley

[3]. *Thinking in JAVA*

Bruce Eckel

Các tài liệu khác

[1]. *www.javabeginner.com*

[2]. *www.javavietnam.org*

[3]. *www.java.sun.com*

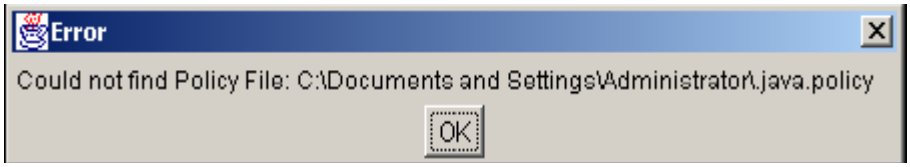
[4]. *www.vi.wikipedia.org*

PHỤ LỤC

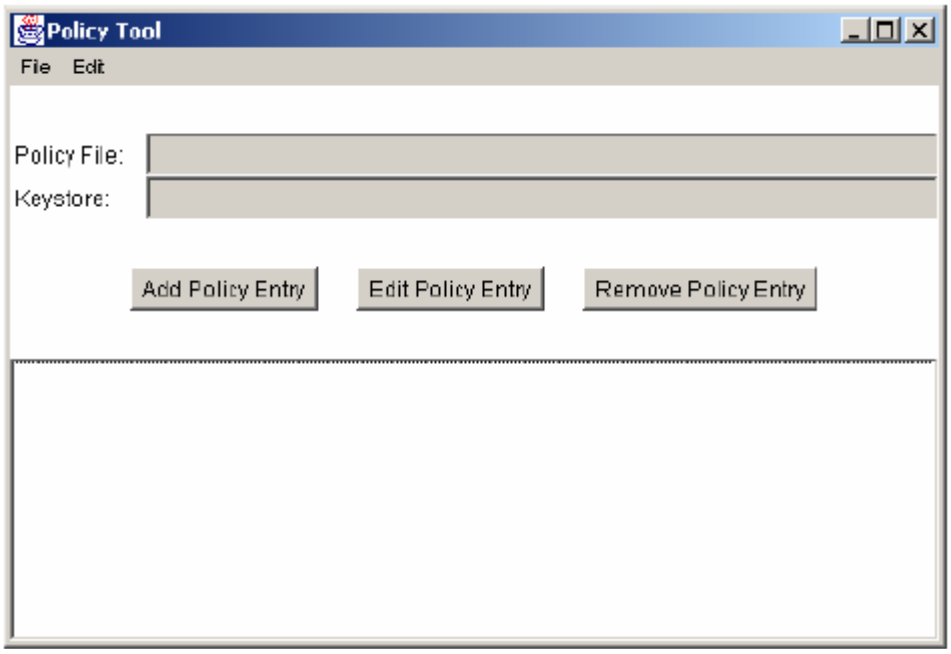
1. Hướng dẫn tạo tệp chính sách `.java.policy`

1.1. Khởi động công cụ tạo tệp chính sách

- Bấm đôi nút chuột trái tại tệp PolicyTool.exe trong thư mục BIN của trình biên dịch JDK. Nhận được thông báo sau trên màn hình

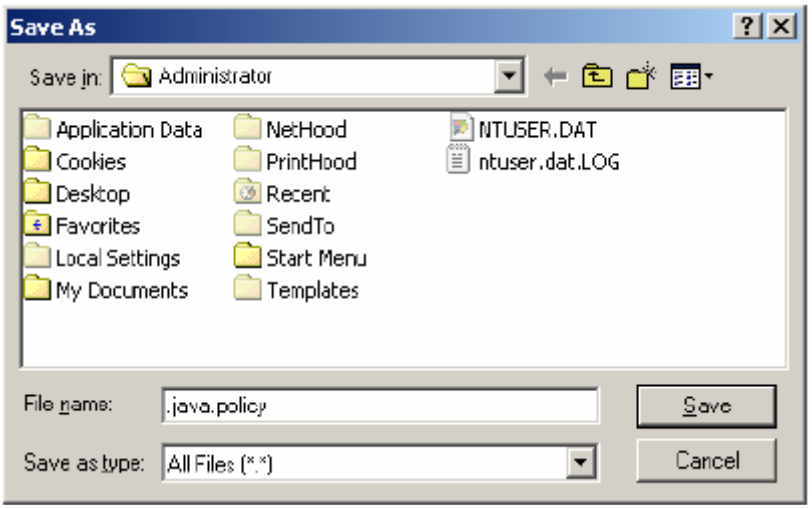


- Đây là lỗi chưa có tệp chính sách (bắt buộc phải đặt tên là `.java.policy`). Chọn OK để đóng cửa sổ thông báo lỗi và trở lại cửa sổ tạo file chính sách như hình dưới

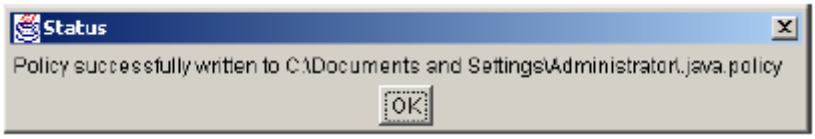


1.2. Tạo file chính sách

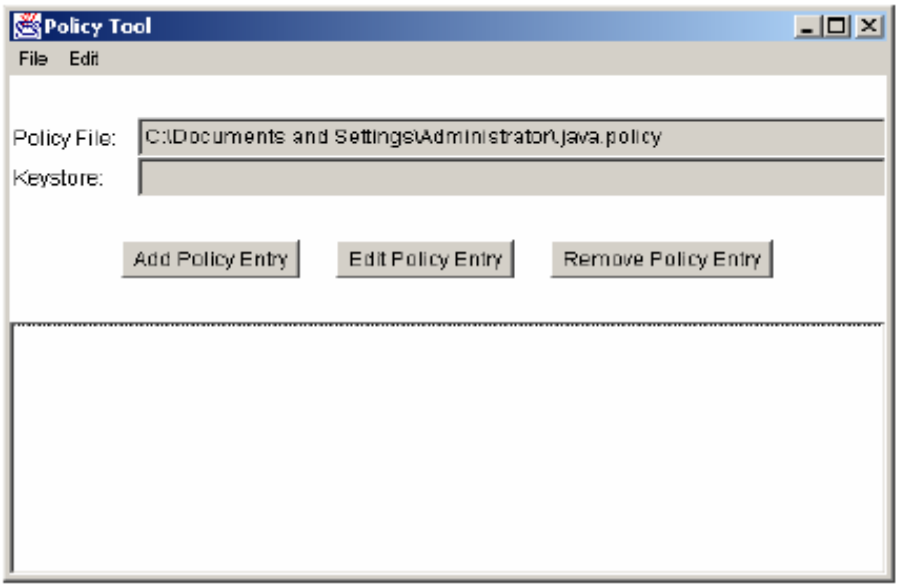
- Mở menu file→Save As. Xuất hiện khung đối thoại



- Chọn thư mục chứa file chính sách theo yêu cầu của Java. Tên file bắt buộc phải gõ là (`.java.policy`). Chọn nút lệnh Save để ghi file lên đĩa. Xuất hiện thông báo

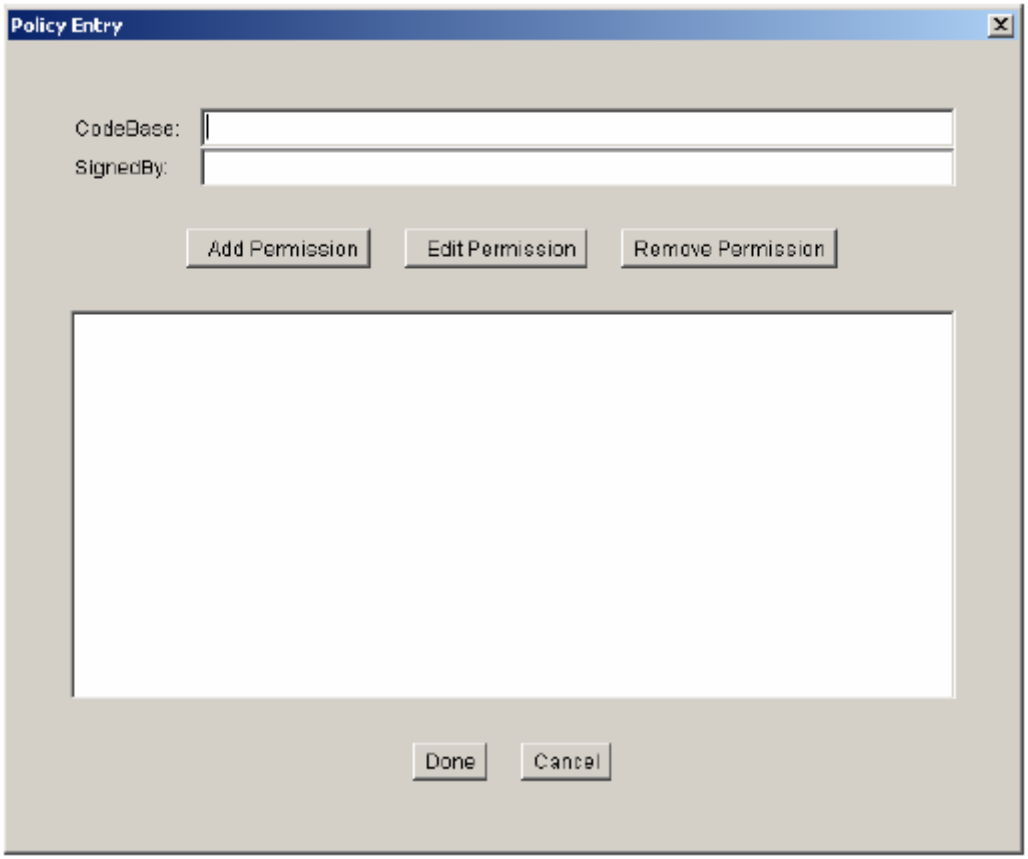


- Chọn **OK** để tiếp tục, lúc này file vẫn rỗng chưa có thông tin. Cửa sổ Policy Tool có dạng như hình dưới

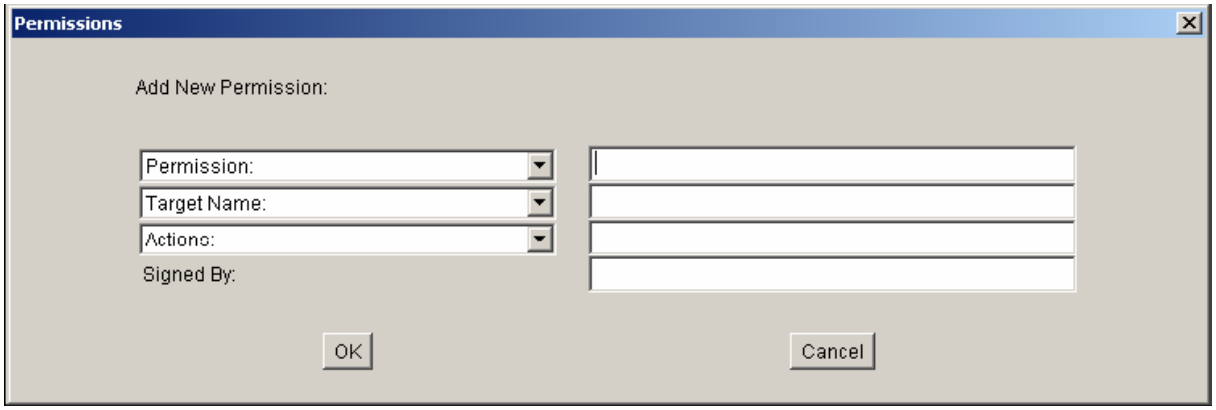


1.3. Cấp quyền sử dụng Java Socket

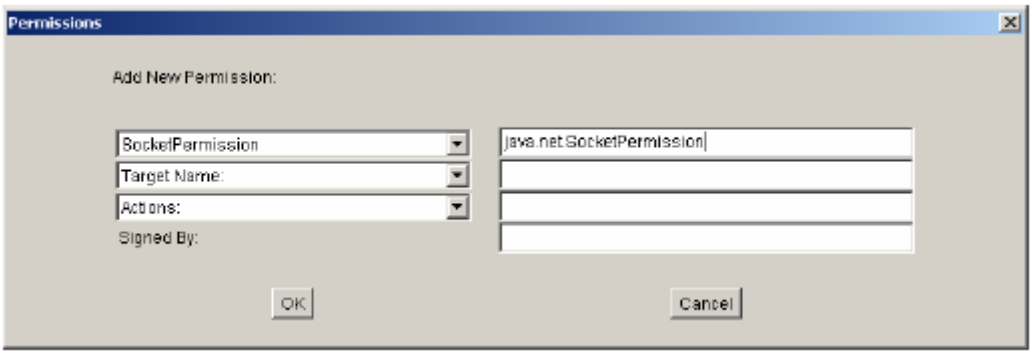
- Bấm chuột trái tại nút lệnh **Add Policy Entry**. Xuất hiện khung đối thoại sau



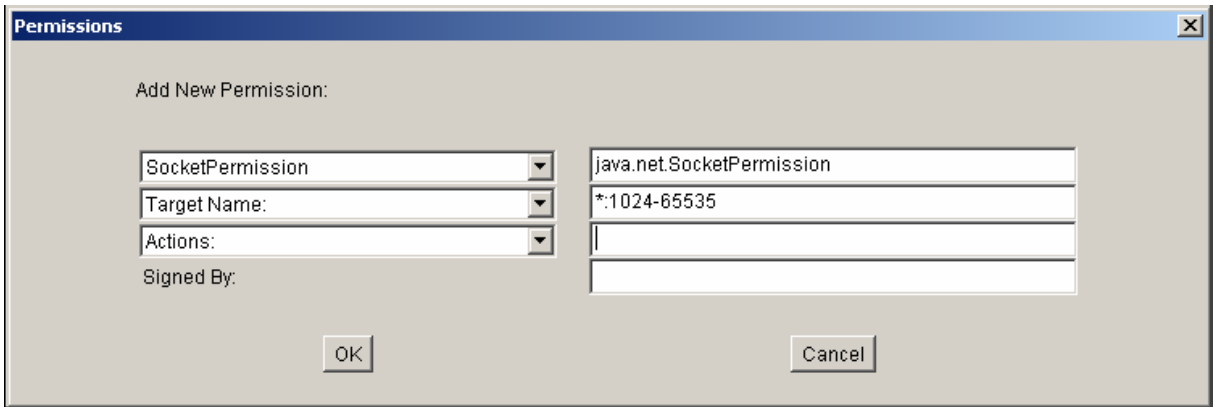
- Bấm đơn nút chuột trái tại nút lệnh **Add Permission**. Xuất hiện khung đối thoại



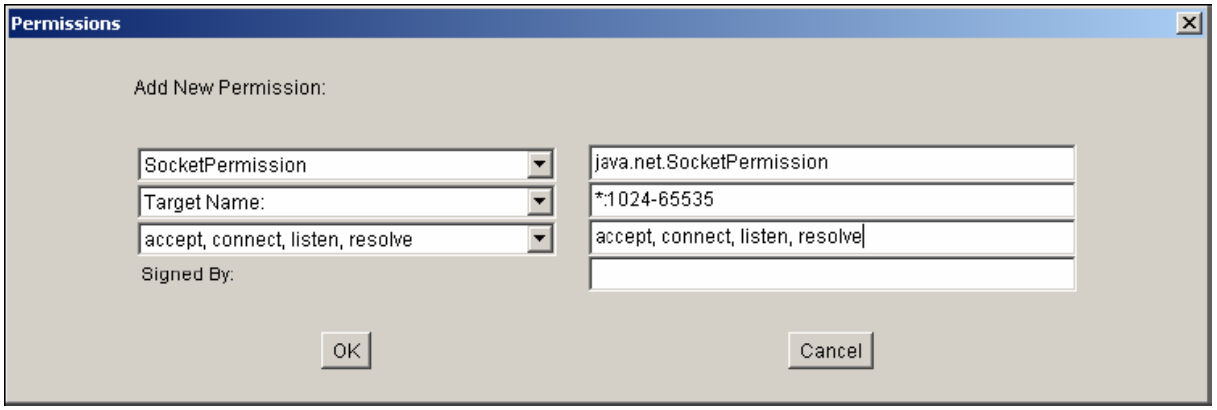
- Trong lựa chọn Permission chọn như hình dưới



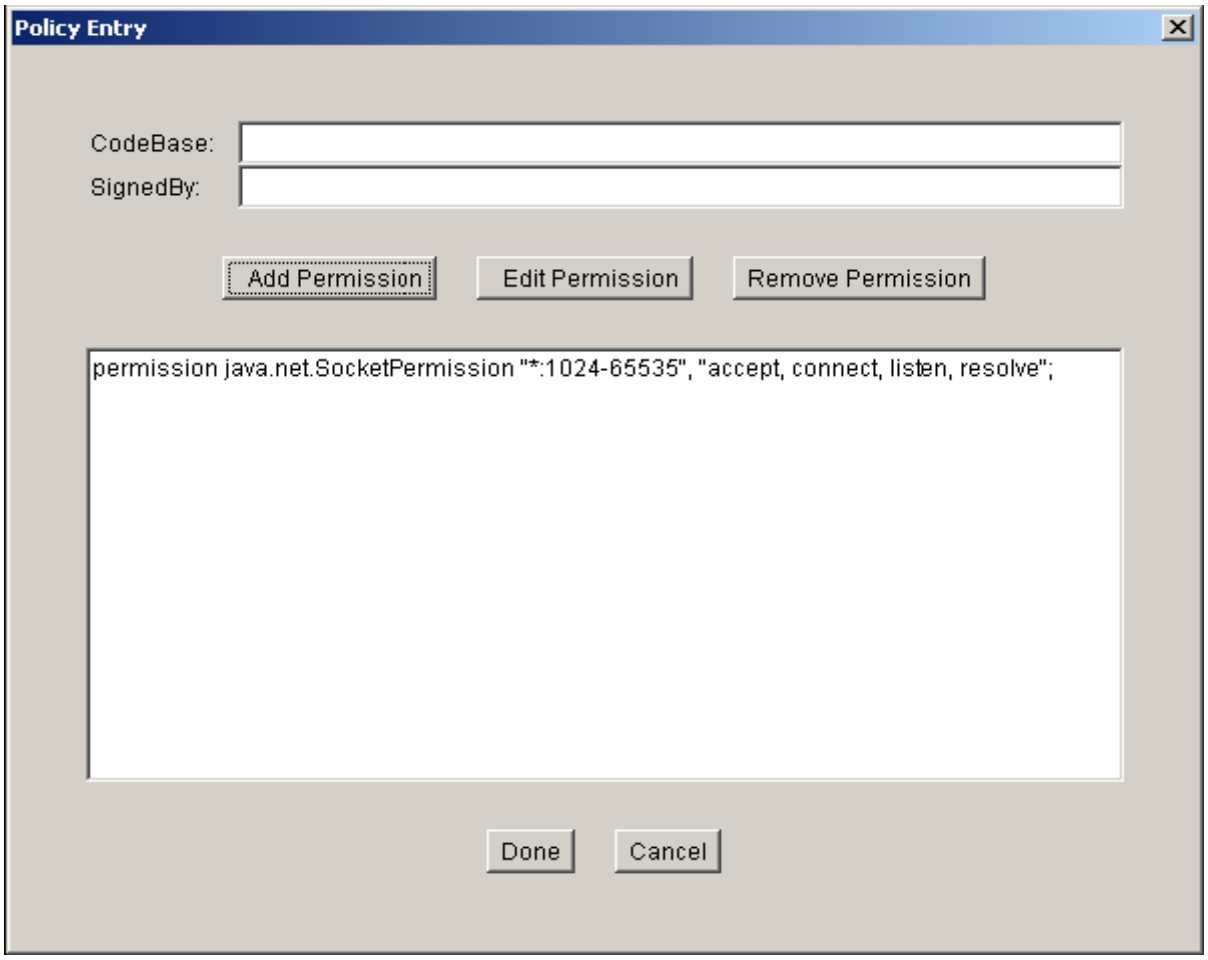
- Trong hộp value của lựa chọn **Target Name** gõ vào giá trị như hình dưới



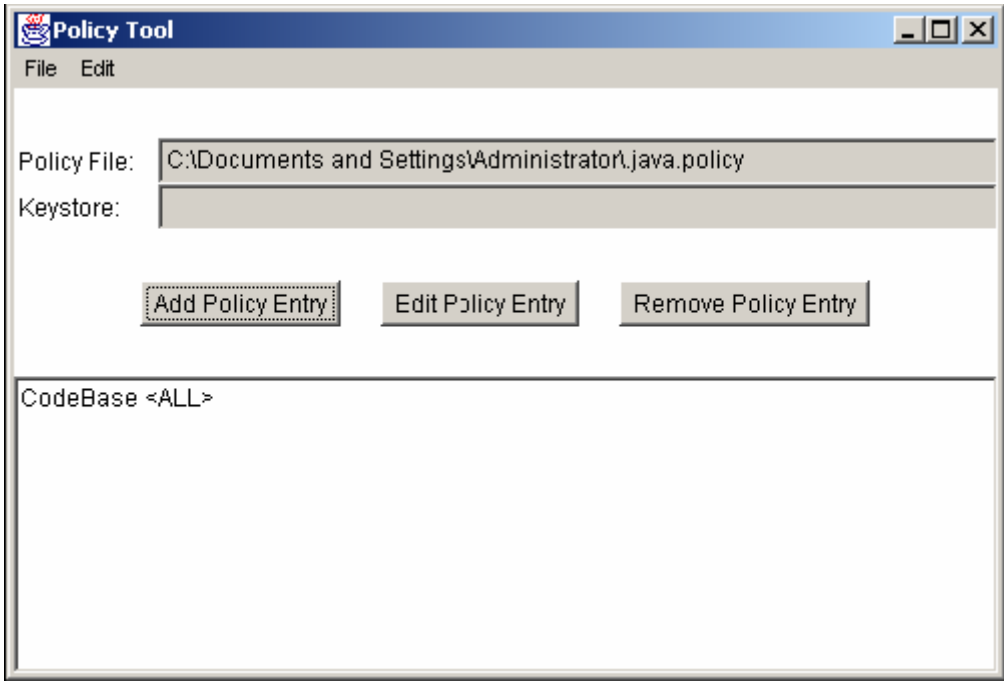
- Trong hộp value của lựa chọn Actions chọn như hình dưới



- Chọn OK kết thúc và quay trở lại khung đối thoại như hình dưới



- Chọn nút lệnh Done kết thúc và quy lại khung đối thoại sau



1.4. Thoát

Mở menu file→ Save để ghi nội dung file chính sách lên đĩa sau đó mở menu file→Exit để thoát.

2. Mã nguồn chương trình

Mô đun phía Server: File `SocketServer.java`

```
//============
import java.net.* ;
import java.io.* ;
import java.sql.* ;
public class SocketServer
{
     static int PORT = 8080 ; // cong mac dinh
     //-------------
     public static void main(String[] arg)
     {
          try
          {
          // Tao socket cho server
            ServerSocket ss = new ServerSocket(PORT);
            System.out.println("Server lang nghe tren port:" +
ss.getLocalPort());
               while(true)
               {
                    try
                    {
                         // lang nghe cac yeu cau ket noi tu Client
                         Socket s = ss.accept();
                         RequestProcessing rp = new RequestProcessing(s);
                         rp.start();
                         // khoi dong phan xu ly cho client hien tai
                    }
                    catch(IOException e)
                    {
                       System.out.println("Connection Error : "+e);
                    }
               }
          }
          catch(IOException e)
          {
                    System.out.println("Create Socket Error : "+e);
          }
     }
}
```

```
//===================================
class RequestProcessing extends Thread
{
     Socket client ; // Socket tra loi cho client
     static String _driver ="com.microsoft.jdbc.sqlserver.SQLServerDriver";
     static                  String                  _url                  =
"jdbc:microsoft:sqlserver://10.5.101.30:1433;DatabaseName=QLSV_H";
     static Connection con ; // doi tuong ket noi
     static Statement stmt ; // dai dien cac cau lenh cua SQL
     static ResultSet r ;   // doi tuong chua ket qua truy van
     //--------------------------------
     public RequestProcessing(Socket s)
     {
          client = s;
     }
     //--------------------------------
     public void run()
     {
          String xau;
          try
          {
               // Nap driver
               Class.forName(_driver) ;
               con =DriverManager.getConnection(_url,"vidu","123") ;
               // thiet lap doi tuong ket noi
          }
          catch(java.lang.ClassNotFoundException e)
          {
           System.out.println("Khong Tim Thay Lop Driver") ;
           System.exit(1) ;
          }
          catch(java.sql.SQLException e)
          {
               System.out.println("Khong Mo Duoc Ket Noi Toi CSDL") ;
               System.exit(1) ;
          }
          try
          {
               // inStream: doi tuong nhan thong diep
```

```
BufferedReader inStream =
     new BufferedReader(
new InputStreamReader(client.getInputStream())) ;

// Tao Ra Doi Tuong Viet Du Lieu Len socket_server
PrintWriter outStream =
     new PrintWriter(
     new BufferedWriter(
new OutputStreamWriter(client.getOutputStream())),true) ;
     boolean finished = false ;
do
{
     String inLine = inStream.readLine() ;
     //chuoi nhan tu clients
     System.out.println("Received : "+inLine) ;
     if(inLine.equalsIgnoreCase("END")) finished = true ;
     try
     {
          stmt = con.createStatement() ;

          if(inLine.equals("*"))
               //in tat ca ban ghi khi nhan ve '*'
          {
               xau="Select
a.masv,a.hoten,a.gioitinh,a.lop,b.tenmon,b.dvht,c.diemthi,c.lanthi     From
tbl_sinhvien as a,tbl_monhoc as b,tbl_diemthi as c where a.masv=c.masv and
b.mamon=c.mamon";
               r = stmt.executeQuery(xau) ; // thuc hien truy
van
          }
          else
          // in ra 1 ban ghi theo masv nhan duoc tu client
               xau="Select
a.masv,a.hoten,a.gioitinh,a.lop,b.tenmon,b.dvht,c.diemthi,c.lanthi     From
tbl_sinhvien as a,tbl_monhoc as b,tbl_diemthi as c where a.masv=c.masv and
b.mamon=c.mamon and a.masv='"+inLine+"'";
               r = stmt.executeQuery(xau);
                    // thuc hien truy van tren
               ResultSetMetaData mrs= r.getMetaData();
                    // cung cap thong tin cau truc cua CSDL
```

```
                    int socot=mrs.getColumnCount();
                              // xac dinh so cot cua mrs

                    String outLine ="" ;
                    for(int j=1;j<= socot;j++)
                    {
                         outLine+=mrs.getColumnName(j)+"\t";
                    }
                    outStream.println(outLine);  // ten cac cot j trong
bang
                    while(r.next())    // doc du lieu cua dong tiep theo
                    {
                         outLine = "" ;
                         for(int i=1; i<=socot; i++)
                         {
                              outLine+= r.getString(i) + "\t";
                                   // du lieu cua cot i
                         }

                              System.out.println("Sent: "+ outLine) ;
                                   outStream.println(outLine) ;

                         } // of while
                              outStream.println("END") ;
                                        // bao hieu het dl gui cho clients
                    } // of try
                    catch(java.sql.SQLException e)
                    {
                         System.err.println("Khong the thuc hien duoc truy
van...") ;
                    }
               }
               while(!finished) ;
               client.close() ;
          }
          catch(IOException e)
          {
               System.out.println(e) ;
          }
     }
}
```

Mô đun phía clients: File `AppletXemDiem.java`

```
import java.applet.Applet ;
import java.awt.* ; // Chua lop ScrollPane
import javax.swing.* ;
import java.awt.event.* ;
import java.net.* ;
import java.io.* ;

public class AppletXemDiem extends Applet implements ActionListener
{
     JTextField txtMasv ;
     JButton btXem ;
     JTextArea taKq ;
     JLabel lbMasv ;
     ScrollPane scroll ;

     Socket connection ;
     PrintWriter out ;
     BufferedReader in ;
//-----------------------------------------
     public void init()
     {
          this.setLayout(null) ;
          Font f = new Font("Times New Roman",Font.PLAIN,12);

          lbMasv = new JLabel("Nhap ma sinh vien") ;
          lbMasv.setBounds(10, 10, 150,25) ;
          add(lbMasv) ;

          txtMasv = new JTextField() ;
          txtMasv.setBounds(150,10,150,25) ;
          txtMasv.addActionListener(this);
          add(txtMasv) ;

          taKq = new JTextArea(460, 300) ;
          scroll = new ScrollPane() ;
          taKq.setFont(f);
          scroll.setBounds(10, 40, 485, 200) ;
          scroll.setEnabled(true) ;
          scroll.add(taKq) ;
          add(scroll) ;
```

```
          btXem = new JButton("Xem") ;
          btXem.setToolTipText("Xem diem mon hoc") ;
          btXem.addActionListener(this) ;
          btXem.setBounds(100, 245, 80, 25) ;
          add(btXem) ;
          try
          {
               // Tao Ket Noi Toi Socket_Server
               connection = new Socket("10.5.101.28",8080) ;
               // Tao Ra Doi Tuong Doc Vung Dem Cua socket_client
               in = new BufferedReader(
                    new InputStreamReader(connection.getInputStream())) ;
               // Tao Ra Doi Tuong Viet Du Lieu Len socket_client
               out = new PrintWriter(
                         new BufferedWriter(
                              new OutputStreamWriter(
                              connection.getOutputStream())),true) ;
          }
          catch(Exception e)
          {
               e.printStackTrace() ;
          }
     }
//---------------------------------
     public void actionPerformed(ActionEvent event)
     {
               String s=txtMasv.getText().trim();
               if(event.getSource() == btXem || event.getSource()==txtMasv)
               {
                    if(txtMasv.getText().length() != 0)
                    {
                         // Gui Request Toi Socket_Server
                         out.println(s) ;
                         // Nhan Du Lieu Gui Tu Socket_Server
                         String st ;
                         taKq.setText("") ;
                         try
                         {
                              while(true)
                              {
                                   st = in.readLine() ;
```

```
                                    if(st.equalsIgnoreCase("END")) break ;
                                    taKq.append(st+"\n ") ;
                               }
                          }
                          catch(IOException e)
                          {
                               System.out.println(e) ;
                          }
                     }
                }
           }
}
```