

LẬP TRÌNH MẠNG

CHƯƠNG I. TỔNG QUAN MẠNG MÁY TÍNH	3
1. Khái niệm & phân loại mạng máy tính	3
1.1. Khái niệm mạng máy tính.....	3
1.2. Phân loại mạng	3
2. Mô hình OSI:	3
2.1. Chức năng mô hình OSI:	3
2.2. Cấu trúc:	3
3. Mô hình TCP/IP	5
3.1. Internet Protocol – IP	5
3.2. Địa chỉ IP.....	5
3.3. Internet Control Message Protocol – ICMP.	6
3.4. Transport layer – TCP	6
3.5. Transport layer – UDP:.....	7
3.6. So sánh TCP vs UDP	8
4. OSI vs TCP/IP.....	9
5. Một số vấn đề khác.....	9
5.1. Default gateway (DG):.....	9
5.2. Chia subnet.....	9
5.4. DNS Server	11
5.5. Router VS Switch * Router:.....	11
5.6. NAT (Network Address Translation)	12
5.7. Open Port:	13
II. LẬP TRÌNH SOCKET	13
1. Khái niệm socket	13
2. Phân loại socket.....	13
3. Lập trình với địa chỉ IP (InetAddress).....	13
4. Lập trình TCPSocket:	14
4.1. Chương trình phía Client	14
4.2. Chương trình phía Server:.....	16
5. Lập trình UPDSocket	20
5.1. Lớp DatagramPacket:	20
5.2. Lớp DatagramSocket:	21
5.3. Xây dựng ứng dụng UDPSocket	22

III. KỸ THUẬT XÂY DỰNG ỨNG DỤNG PHÍA SERVER	23
1. Giới thiệu	23
2. Concurrent TCP Server:	23
3. Tạo Thread:	24
4. ThreadPool:	26
5. Executor Framework:	26
6. Ví dụ ThreadPoolExecutor:	29
7. Ví dụ TCPSocket multithread	30
BÀI TẬP	32
1. ChatRoom TCP:	32
2. Giải phương trình bậc 1	35
3. Giải phương trình bậc 2 – UPD	37
5. Database Connection	44
6. Xử lý số:.....	44
7. Từ điển	50
8. Chuỗi đảo ngược TCP	53
9. Số hoàn hảo TCP	55
11. Viết chương trình đoán số, hoạt động theo mô hình client-server, sử dụng TCP socket	59
14. Viết chương trình tra cứu thông tin cá nhân, hoạt động theo mô hình client-server, sử dụng TCP socket	66
15. Viết chương trình tính toán, hoạt động theo mô hình client-server, sử dụng TCP socket	68
16. Viết chương trình tra thông tin và điểm sinh viên trên cổng thông tin đào tạo, sử dụng mô hình client-server, dựa trên TCPSocket.....	70
17. Viết chương trình gửi tin nhắn hai chiều (chuỗi đảo ngược) giữa client-server sử dụng UDP socket.	73

CHƯƠNG I. TỔNG QUAN MẠNG MÁY TÍNH

1. Khái niệm & phân loại mạng máy tính

1.1. Khái niệm mạng máy tính

- Là một tập hợp các máy tính được nối với nhau bởi đường truyền theo một cấu trúc nào đó và qua đó, chúng trao đổi thông tin qua lại với nhau.
- Đường truyền dữ liệu tạo nên cấu trúc mạng

1.2. Phân loại mạng

Dựa trên :

- Khoảng cách địa lý
- Phương tiện kết nối
- Mô hình mạng: client-server; peer-to-peer

Ví dụ: Mạng cục bộ (LAN), mạng diện rộng (WAN), mạng đô thị (MAN)

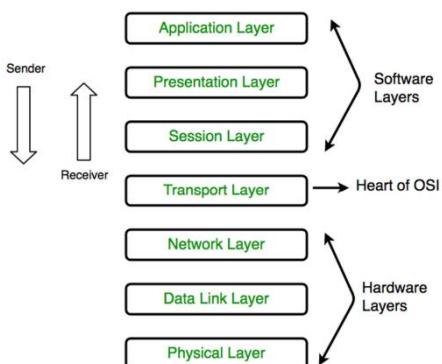
2. Mô hình OSI:

Mô hình tham chiếu kết nối các hệ thống mở - được Tổ chức quốc tế về chuẩn hóa ISO đưa ra nhằm cung cấp một mô hình chuẩn cho các nhà sản xuất và cung cấp sản phẩm viễn thông áp dụng theo để phát triển các sản phẩm viễn thông. Ý tưởng mô hình hóa được tạo ra còn nhằm hỗ trợ cho việc kết nối giữa các hệ thống và module hóa các thành phần phục vụ mạng viễn thông.

2.1. Chức năng mô hình OSI:

- Cung cấp kiến trúc về hoạt động của kết nối liên mạng.
- Đưa ra trình tự công việc để thiết lập và thực hiện một giao thức cho kết nối các thiết bị trên mạng.
- Chia nhỏ các hoạt động phức tạp của mạng thành các phần công việc đơn giản.
- Cho phép các nhà thiết kế có khả năng phát triển trên từng module chức năng.
- Cung cấp các khả năng định nghĩa các chuẩn giao tiếp có tính tương thích cao “plug and play” và tích hợp nhiều nhà cung cấp sản phẩm.

2.2. Cấu trúc:



Physical:

- Chuyển bit thô qua phương tiện truyền dẫn.
- Đơn vị dữ liệu: Bits.
- Các thiết bị bao gồm: Hub, Repeater, Cables.

Data Link:

- Đảm bảo dữ liệu truyền tin cậy từ một node đến một node khác qua kênh truyền vật lý. Hai tầng con: Logical Link Control (LLC) & Media Access Control (MAC)

- Đơn vị dữ liệu: Frames.
- Các thiết bị bao gồm: NIC, Switch, Bridge.

- Functions:
 - + Framing.
 - + Physical addressing.
 - + Error control.
 - + Flow Control.
 - + Access control.

Network:

- Truyền dữ liệu giữa các host ở các mạng khác nhau (mạng logic), packet routing.
- Đơn vị dữ liệu: Packet.
- Router.
- Functions:
 - + Routing
 - + Logical Addressing

Transport

- Nhận thông tin từ tầng Session chia thành các gói nhỏ hơn và truyền xuống lớp dưới, hoặc nhận thông tin từ lớp dưới chuyển lên phục hồi theo cách chia của hệ phát.
- Đơn vị dữ liệu: Segment.
- Functions:
 - + Segmentation and Reassembly.
 - + Service Point Addressing.

Session:

- Thiết lập kết nối, duy trì session.
- Đơn vị thông tin: Data
- Functions:
 - + Session establishment, maintenance and termination
 - + Synchronization

Presentation (Translation):

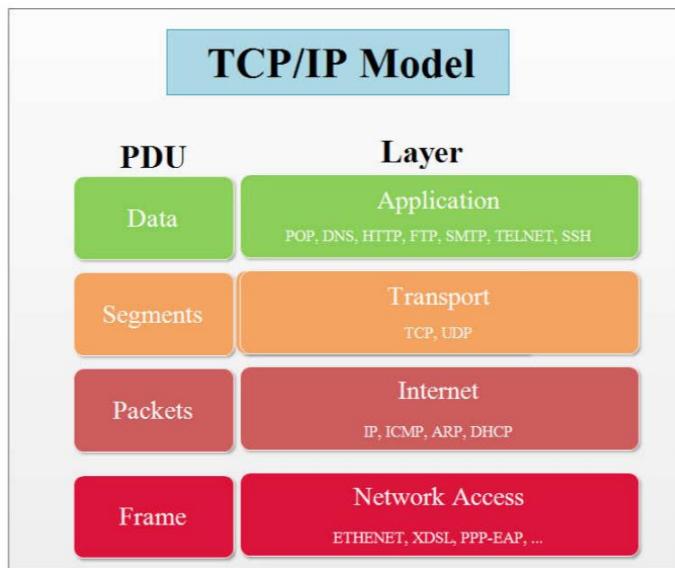
- Xác lập dạng thức dữ liệu được trao đổi.
- Đơn vị dữ liệu: Data
- Functions:
 - + Translation (ex ASCII to EBCDIC)
 - + Encryption/ Decryption
 - + Compression

Application:

- Giao diện giữa các chương trình ứng dụng của người dùng và mạng. Cung cấp các dịch vụ mạng.
- Đơn vị dữ liệu: Data.
- Functions:
 - + Network Virtual Terminal.
 - + FTAM-File transfer access and management.
 - + Mail Services.
 - + Directory Services

3. Mô hình TCP/IP

Phát triển bởi DARPA, đưa thành chuẩn vào 1983. Là viết tắt của Transmission Control Protocol/Internet Protocol – Giao thức điều khiển truyền nhận/ Giao thức liên mạng. Đây là một bộ các giao thức truyền thông được sử dụng để kết nối các thiết bị mạng với nhau trên internet. TCP/IP cũng có thể được sử dụng như một giao thức truyền thông trong mạng máy tính riêng (mạng nội bộ).



3.1. Internet Protocol – IP

- Giao thức nền tảng tạo nên mạng Internet.
- Chức năng:
 - + Định nghĩa cấu trúc các gói dữ liệu là đơn vị cơ sở cho việc truyền dữ liệu trên Internet.
 - + Định tuyến để chuyển các gói dữ liệu.
- Đặc tính:
 - + Có khả năng phát hiện lỗi trong phần header.
 - + Best-effort: không tin cậy và không có kết nối.
 - + Phân mảnh & hợp nhất: MTU (Maximum Transfer Unit)

3.2. Địa chỉ IP

- Là một địa chỉ đơn nhất mà những thiết bị điện tử hiện nay đang sử dụng để nhận diện và liên lạc với nhau trên mạng máy tính bằng cách sử dụng giao thức Internet.
- IPv4 có chiều dài 4 bytes tương ứng 32 bits để đánh địa chỉ. Gồm hai phần là NetworkID và HostID. Dùng netmask để phân bố số bit dùng cho NetworkID.
- Bao gồm 5 lớp mạng (A, B, C, D và E). Lớp A sử dụng 8 bits cho phần network, do đó có tới 24 bits được sử dụng cho phần host. Lớp B dùng 16 bits cho network, 16 bit dành cho host. 24 bits được sử dụng để xác định phần network cho lớp C, do đó, mỗi network của lớp C chỉ còn 8 bit để đánh địa chỉ host. Lớp D được dùng cho địa chỉ multicast còn lớp E sử dụng cho thí nghiệm.
- Public IP: Địa chỉ IP mà qua đó thiết bị giao tiếp với các thiết bị khác trên Internet.
- Private IP: Địa chỉ IP cá nhân (hay còn gọi là IP Private) là địa chỉ riêng sử dụng trong nội bộ mạng LAN như mạng gia đình, nhà trường, công ty. Khác với IP Public, IP Private không thể kết nối với mạng Internet mà chỉ có các thiết bị trong mạng mới có thể giao tiếp với nhau thông qua bộ định tuyến router. Địa chỉ IP riêng được bộ định tuyến gán tự động hoặc bạn có thể tự thiết lập lại theo cách thủ công. Ba dải địa chỉ dùng cho Private IP:

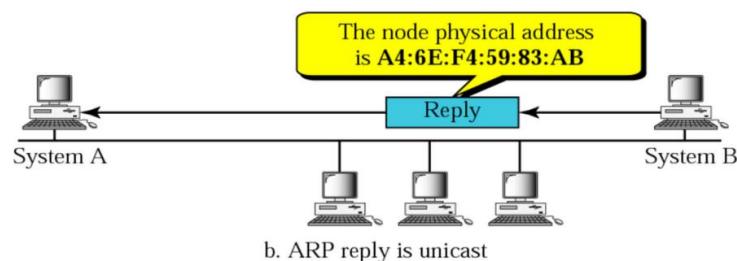
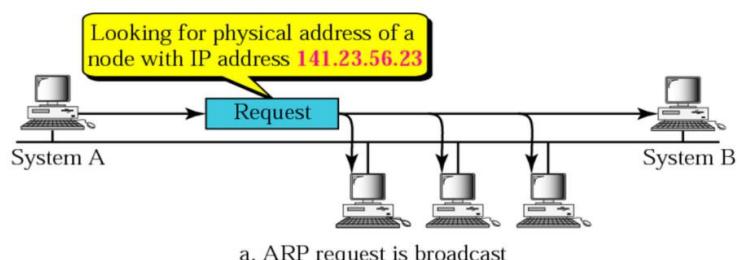
Tên	Dài địa chỉ	Số lượng địa chỉ trong dài	Mô tả mạng đầy đủ	Khối CIDR lớn nhất
Khối 24-bit	10.0.0.0–10.255.255.255	16.777.216	Một dải trọn vẹn thuộc lớp A	10.0.0.0/8
Khối 20-bit	172.16.0.0–172.31.255.255	1.048.576	Tổ hợp từ mạng lớp B	172.16.0.0/16
Khối 16-bit	192.168.0.0–192.168.255.255	65.536	Tổ hợp từ mạng lớp C	192.168.0.0/24

□ IP address VS MAC address

Address Class	1st octet range (decimal)	1st octet bits (green bits do not change)	Network(N) and Host(H) parts of address	Default subnet mask (decimal and binary)	Number of possible networks and hosts per network
A	1-127**	00000000-01111111	N.H.H.H	255.0.0.0	128 nets (2^7) 16,777,214 hosts per net (2^{24-2})
B	128-191	10000000-10111111	N.N.H.H	255.255.0.0	16,384 nets (2^{14}) 65,534 hosts per net (2^{16-2})
C	192-223	11000000-11011111	N.N.N.H	255.255.255.0	2,097,150 nets (2^{21}) 254 hosts per net (2^{8-2})
D	224-239	11100000-11101111	NA (multicast)		
E	240-255	11110000-11111111	NA (experimental)		

3.3. Internet Control Message Protocol – ICMP.

- Internet Control Message Protocol (ICMP): gửi các thông báo lỗi và các thông báo điều khiển.
- Address Resolution Protocol (ARP): ánh xạ từ địa chỉ IP thành MAC.



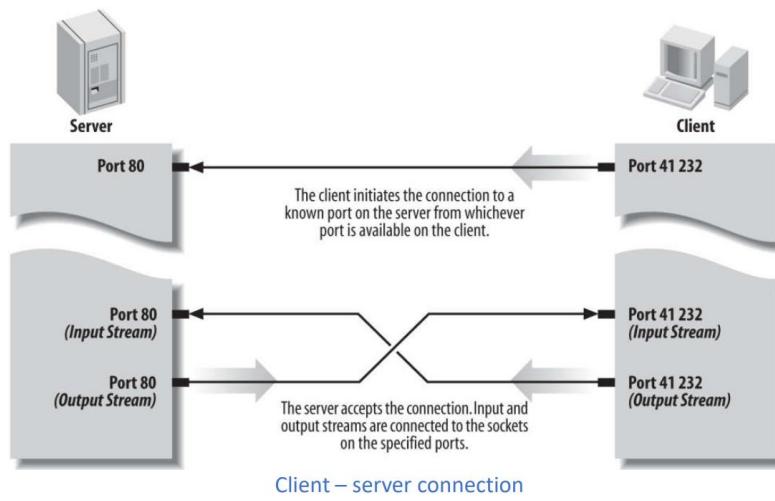
3.4. Transport layer – TCP

Transmission Control Protocol (TCP) là giao thức tiêu chuẩn trên Internet đảm bảo trao đổi thành công các gói dữ liệu giữa các thiết bị qua mạng. TCP là giao thức truyền tải hướng kết nối, nghĩa là phải thực hiện thiết lập kết nối với đầu xa trước khi thực hiện truyền dữ liệu. Tiến trình thiết lập kết nối ở TCP được gọi là tiến trình bắt tay 3 bước.

TCP có độ tin cậy cao, hoạt động bằng cách phân mảnh thông điệp và ráp lại ở đích.

* Địa chỉ ứng dụng:

- Port number: mỗi ứng dụng được gán một chỉ số nguyên.
- Mô hình client-server:
 - + Server: dùng well-known port.
 - + Client: lấy port còn trống từ hệ thống.
- Port:
 - + Well-known: 0 -> 1023
 - + Registered: 1024 -> 49151
 - + Dynamic/private: 49152 -> 65535



2

Transmission Control Protocol (TCP) Header 20-60 bytes

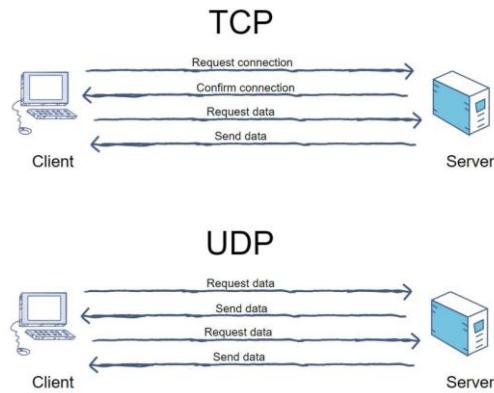
source port number 2 bytes	destination port number 2 bytes
sequence number 4 bytes	
acknowledgement number 4 bytes	
data offset 4 bits	reserved 3 bits
control flags 9 bits	
	window size 2 bytes
checksum 2 bytes	urgent pointer 2 bytes
optional data 0-40 bytes	

3.5. Transport layer – UDP:

UDP là giao thức truyền tải hướng không kết nối (connectionless). Nó sẽ không thực hiện thao tác xây dựng kết nối trước khi truyền dữ liệu mà thực hiện truyền ngay lập tức khi có dữ liệu cần truyền.

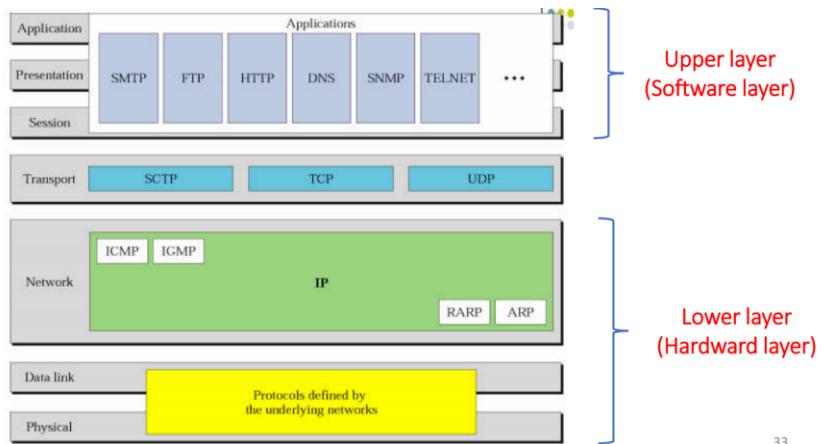
UDP không đảm bảo tính tin cậy khi truyền dữ liệu, có khả năng phát hiện lỗi, không điều khiển dòng, điều khiển lỗi (window, ACK) và sử dụng port number.

3.6. So sánh TCP vs UDP



TCP	UDP
Hướng kết nối	Hướng không kết nối
Độ tin cậy cao	Độ tin cậy thấp
Tốc độ truyền thấp hơn UDP	Tốc độ truyền cao
Không cho phép mất gói tin	Cho phép mất gói tin
Đảm bảo việc truyền dữ liệu	Không đảm bảo việc truyền dữ liệu
Có sắp xếp thứ tự các gói tin	Không sắp xếp thứ tự các gói tin
Gửi dữ liệu dạng luồng byte	Gửi đi datagram
Có cơ chế khôi phục lỗi	Không có cơ chế khôi phục lỗi
Kiểm soát luồng	Không kiểm soát luồng
Header 20 byte cho phép	Header 8 byte chỉ cho phép dữ liệu bắt buộc.

4. OSI vs TCP/IP



33

Nội dung	OSI	TCP/IP
Độ tin cậy và phổ biến	Hạn chế	Được chuẩn hóa, nhiều người tin cậy và sử dụng phổ biến trên toàn cầu
Phương pháp tiếp cận	Tiếp cận theo chiều dọc	Tiếp cận theo chiều ngang
Sự kết hợp giữa các tầng	Mỗi tầng khác nhau sẽ thực hiện một nhiệm vụ khác nhau, không có sự kết hợp giữa bất cứ tầng nào	Trong tầng ứng dụng có tầng trình diễn và tầng phiên được kết hợp với nhau
Thiết kế	Phát triển mô hình trước sau đó sẽ phát triển giao thức	Các giao thức được thiết kế trước sau đó phát triển mô hình
Số lớp (tầng)	7	4
Truyền thông	Hỗ trợ cả kết nối định tuyến và không dây	Hỗ trợ truyền thông không kết nối từ tầng mạng
Tính phụ thuộc	Giao thức độc lập	Phụ thuộc vào giao thức

5. Một số vấn đề khác

5.1. Default gateway (DG):

Bản chất là một địa chỉ IP, và còn được gọi là cổng mặc định của mạng máy tính. Địa chỉ này sẽ được cấu hình cho thiết bị máy tính và máy tính mặc định sẽ gửi gói tin đến địa chỉ này để có thể tiếp tục đi đến nơi khác.

5.2. Chia subnet

Địa chỉ IP bao gồm 2 phần: host address và network address được chia bởi subnet mask. Subnet là một số dạng 32 bit. Để tạo ra Subnet mask, người ta sẽ đặt host bit dưới dạng số 0 và network bit dạng số 1. Từ đó tạo ra các dãy số có dạng nhị phân là 0 và 1 để phân chia địa chỉ IP thành 2 phần, tương ứng với địa chỉ mạng và địa chỉ host.

5.2.1. Công thức tính

Gọi n là số bit 1 tăng thêm của Subnet Mask (hay còn gọi là số bit mượn).

Gọi m là số bit 0 còn lại của Subnet Mask ($m = 32 - n - \text{SM hiện tại}$).

Ta làm theo 5 bước sau:

- Bước 1: Số Subnet: 2^n
- Bước 2: Số Host/Subnet : $2^m - 2$ (vì phải trừ đi địa chỉ NetID và Broadcast)
- Bước 3: Bước nhảy: 2^m
- Bước 4: Subnet mask mới: $256 - \text{Bước nhảy}$
- Bước 5: Các Subnet ID gồm:
 - + Subnet ID đầu tiên = 0
 - + Subnet ID kế tiếp = Subnet hiện tại + Bước nhảy
- Bước 6: Trong Subnet ID:
 - + Host đầu: Subnet ID + 1
 - + Host cuối: Subnet ID + Bước nhảy - 2
 - + Địa chỉ Broadcast: Host cuối + 1

5.2.2. Bài tập chia subnet 1

Ta phải chia Net ID: 203.162.4.0/24 tăng 2 bit ($n = 2$)

1. Số Subnet: $2^n = 2^2 = 4$
2. Số Host trên Subnet : $2^6 - 2 = 62$
3. Bước nhảy: $2^6 = 64$
4. Subnet mask mới: $256 - \text{Bước nhảy} = 256 - 64 = 192$

Subnet mới: $255.255.255.192 = 1111111.1111111.1111111.11000000 \Rightarrow /26$

5. Các Subnet ID:

Subnet ID đầu tiên = 0 --> 203.162.4.0/26

Subnet ID kế tiếp = Subnet hiện tại + Bước nhảy

203.162.4.64/26

203.162.4.128/26

203.162.4.192/26

Subnet ID	Host đầu: Subnet ID + 1	Host cuối: Subnet ID + Bước nhảy - 2	Broadcast: Host cuối + 1
203.162.4.0/26	203.162.4.1	203.162.4.62	203.162.4.63
203.162.4.64/26	203.162.4.65	203.162.4.126	203.162.4.127
203.162.4.128/26	203.162.4.129	203.162.4.190	203.162.4.191
203.162.4.192/26	203.162.4.193	203.162.4.254	203.162.4.255

5.2.3. Bài tập chia subnet số 2:

Ta có địa chỉ của 1 host, vậy làm sao để suy ra được host đó thuộc vùng mạng (Net ID) nào?

Ví dụ ta có 1 host như sau:

IP: 203.162.4.165

Subnet Mask: 255.255.255.224

Ta thấy giá trị SM: $255.255.255.224 = 11111111 . 11111111 . 11111111 . 11100000$

--> Số bit 0 còn lại của SM là: $m = 5$

--> Bước nhảy = $2^m = 2^5 = 32$

--> Ta lấy $165 : 32 = 5,15625$

--> Ta lấy phần nguyên của kết quả trên tức là $5 \times 32 = 160$

--> Host trên thuộc Net ID: 203.162.4.160

5.4. DNS Server

DNS Server, với tên gọi đầy đủ là Domain Name System được hiểu là một hệ thống phân giải tên miền trên Internet. DNS Server là máy chủ chứa cơ sở dữ liệu về địa chỉ IP public và các hostname được liên kết với chúng, biến đổi tên miền thành địa chỉ IP và trả đến trang web được yêu cầu.

5.5. Router VS Switch

* Router:

Router (Bộ định tuyến) là thiết bị mạng máy tính dùng để chuyển các gói dữ liệu qua một liên mạng và đến các đầu cuối, thông qua một tiến trình gọi là "định tuyến". Nó hoạt động ở tầng thứ 3 (Tầng giao vận) theo mô hình OSI.

* Switch

Switch (thiết bị chuyển mạch) là một thiết bị dùng để kết nối các đoạn mạng với nhau theo mô hình mạng hình sao (star). Theo mô hình này, switch đóng vai trò là thiết bị trung tâm, tất cả các máy tính đều được nối về đây. Trong mô hình tham chiếu OSI, switch hoạt động ở tầng liên kết dữ liệu, ngoài ra có một số loại switch cao cấp hoạt động ở tầng mạng.

So sánh:

Router	Switch
Nó truyền dữ liệu dưới dạng các gói.	Nó truyền dữ liệu dưới dạng khung và gói
Nó sử dụng địa chỉ IP cho mục đích truyền dữ liệu.	Nó sử dụng địa chỉ MAC cho mục đích truyền dữ liệu.
Nó sử dụng lớp 3 của mô hình OSI. Lớp 3 là lớp mạng.	Nó sử dụng lớp 2 của mô hình OSI. Lớp 2 là lớp liên kết dữ liệu.
Bộ định tuyến chứa 2 cổng theo mặc định như cổng Ethernet nhanh. Tuy nhiên, chúng	Ngược lại, các cổng tắc có sẵn với các cổng khác nhau - 8, 16, 24, 48 và 64.

ta có thể thêm các cổng nối tiếp một cách rõ ràng.	
Nó sử dụng bảng định tuyến cho các tuyến để đến IP đích.	Nó sử dụng bảng CAM (Bộ nhớ địa chỉ nội dung) cho địa chỉ MAC.
Bộ định tuyến trong mạng được sử dụng để kết nối hai Mạng khác nhau	Nó được sử dụng để kết nối các thiết bị End như máy tính, máy in, máy quét, v.v.
Nó được sử dụng cho cả mạng WAN / LAN/MAN	Nó chỉ được sử dụng cho các mạng LAN.
Theo mặc định, bộ định tuyến ở chế độ song công toàn phần	Song công toàn phần/bán song công (half duplex)
Trong bộ định tuyến, có thể thực hiện dịch địa chỉ Mạng cũng như dịch địa chỉ cổng	Trong Switch, không thể thực hiện dịch địa chỉ mạng cũng như dịch địa chỉ cổng.

5.6. NAT (Network Address Translation)

NAT (Network Address Translation) là một kỹ thuật cho phép chuyển đổi từ một địa chỉ IP này thành một địa chỉ IP khác. Thông thường, NAT được dùng phổ biến trong mạng sử dụng địa chỉ cục bộ, cần truy cập đến mạng công cộng (Internet). Vị trí thực hiện NAT là router biên kết nối giữa hai mạng.

NAT gồm có 3 loại:

- NAT tĩnh (static NAT): Là phương thức NAT one-to-one. Nghĩa là một địa chỉ IP cố định trong LAN sẽ được ánh xạ ra một địa chỉ IP Public cố định trước khi gói tin đi ra Internet. Static NAT sẽ rất hữu ích trong trường hợp các thiết bị có địa chỉ cố định để truy cập internet từ bên ngoài.
- NAT động (dynamic NAT): là một giải pháp tiết kiệm IP Public cho NAT tĩnh. Thay vì ánh xạ từng IP cố định trong LAN ra từng IP Public cố định. LAN động cho phép NAT cả dải IP trong LAN ra một dải IP Public cố định ra bên ngoài.
- NAT Overload (PAT): Là giải pháp được dùng nhiều nhất. PAT là kết hợp IP Public và số hiệu cổng (port) trước khi đi ra Internet. Lúc này mỗi IP trong LAN khi đi ra Internet sẽ được ánh xạ ra một IP Public kết hợp với số hiệu cổng.

Lợi ích của NAT:

- Giúp tái sử dụng địa chỉ IP, tiết kiệm lượng địa chỉ IP riêng.
- Cung cấp các tính năng bảo mật tốt giúp tăng cường bảo mật cho các mạng riêng bằng cách tách mạng nội bộ khỏi mạng bên ngoài.
- Giúp bảo tồn không gian địa chỉ IP. Bạn có thể kết nối một số lượng lớn máy chủ lưu trữ bằng địa chỉ IP nhỏ với internet toàn cầu.

Hạn chế:

- NAT tiêu thụ tài nguyên bộ nhớ, vì NAT cần dịch địa chỉ IPv4 cho tất cả các biểu đồ dữ liệu IPv4 từ hệ thống nội bộ ra ngoài và ngược lại đồng thời để giữ các chi tiết dịch trong bộ nhớ.
- Có thể gây ra sự chậm trễ trong giao tiếp IPv4.
- Gây mất khả năng truy xuất nguồn gốc IP của thiết bị đầu cuối
- Một số công nghệ và ứng dụng mạng sẽ không hoạt động như mong đợi trong hệ thống mạng được cấu hình NAT.

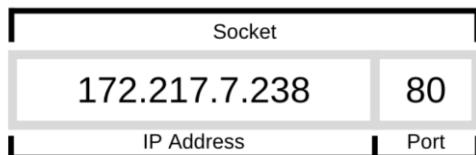
5.7. Open Port:

Là chúng ta cho phép cổng đó được thiết bị (hay địa chỉ IP của thiết bị đó) sử dụng để trao đổi dữ liệu và truy cập ra vào trong hệ thống mạng.

II. LẬP TRÌNH SOCKET

1. Khái niệm socket

- Socket: là điểm cuối (end point) của một liên kết truyền thông 2 chiều giữa 2 chương trình chạy trên môi trường mạng internet.
- Socket: “cửa” nằm giữa process ứng dụng và end-end transport protocol (TCP, UDP).
- Góc độ mạng: Socket là 1 trong 2 điểm cuối của đường nối kết 2 chiều giữa 2 chương trình thực thi trên mạng
- Góc độ người lập trình: Socket là giao diện lập trình ứng dụng (API) hay bộ thư viện hàm hỗ trợ, dùng để nối kết chương trình ứng dụng với lớp mạng trong hệ thống mạng TCP/IP.
- Mỗi tiến trình khi muốn truyền thông bằng socket phải tạo ra một socket và socket đó phải được gán một định danh duy nhất được gọi là địa chỉ socket. Socket = IP + Port => Địa chỉ socket xác định một đầu mút cuối truyền thông. Nó chỉ ra tiến trình truyền thông nào(port) và chạy trên máy nào(IP) sẽ thực hiện truyền thông.



2. Phân loại socket

- Stream socket: Đây là một socket hướng kết nối hoạt động qua giao thức TCP. Có nghĩa là nó chỉ hoạt động khi máy chủ và client đã kết nối thành công với nhau.
- Datagram socket: Datagram Socket là Socket không hướng kết nối và chúng hoạt động qua giao thức UDP (viết tắt của User Datagram Protocol). Vì thế, Socket này có thể hoạt động dù không có bất kỳ sự kết nối nào của 2 máy với nhau.
- Raw socket: Cho phép truyền thông đến các giao thức ở tầng mạng thấp hơn cả tầng transport (VD: giao thức ICMP của tầng Internet)

3. Lập trình với địa chỉ IP (InetAddress)

Lớp InetAddress

- Lớp mô tả về địa chỉ IP
- Không có constructor => không tạo đối tượng InetAddress bằng toán tử new
- Phương thức getLocalHost, getByName, hay getAllByName để tạo một InetAddress instance.

- ✓ `public static InetAddress InetAddress.getByName(String hostname)`
- ✓ `public static InetAddress [] InetAddress.getAllByName(String hostname)`
- ✓ `public static InetAddress InetAddress.getLocalHost()`



10

```

try {
    InetAddress add = InetAddress.getByName("sgu.edu.vn");
    System.out.println(add.getHostAddress());
} catch (UnknownHostException e) {
    System.out.println("Could not find sgu.edu.vn");
}
  
```

```

Console >
<terminated> Example [Java Application] C:\Program Files\Ja
124.158.13.4
  
```

Lấy địa chỉ IP tương ứng với domain sgu.edu.vn

```

try {
    InetAddress[] add = InetAddress.getAllByName("google.com");
    for(int i=0; i<add.length; i++)
        System.out.println(add[i]);
} catch (UnknownHostException e) {
    System.out.println("Could not find sgu.edu.vn");
}
  
```

```

Console >
<terminated> Example [Java Application] C:\Program Files\U
google.com/74.125.24.102
google.com/74.125.24.100
google.com/74.125.24.113
google.com/74.125.24.101
google.com/74.125.24.139
google.com/74.125.24.138
  
```

Lấy tất cả địa chỉ IP tương ứng với domain google.com

```

try {
    InetAddress myAdd = InetAddress.getLocalHost();
    System.out.println(myAdd.getHostName());
    System.out.println(myAdd.getHostAddress());
} catch (UnknownHostException e) {
    System.out.println("Could not find localhost");
}
  
```

```

Console >
<terminated> Example [Java Application] C:\Program Files\U
PG
192.168.56.1
  
```

Lấy tất cả địa chỉ IP tương ứng với domain google.com

4. Lập trình TCPSocket:

4.1. Chương trình phía Client

Lớp Socket: Dùng để tạo đối tượng socket cho phép truyền thông với giao thức TCP hoặc UDP. (giao thức UDP thường sử dụng lớp DatagramSocket thay vì lớp Socket).

Constructor:

```
✓ public Socket(String host, int port) throws UnknownHostException, IOException  
✓ public Socket(InetAddress host, int port) throws IOException  
✓ public Socket(String host, int port, InetAddress interface, int localPort) throws  
IOException, UnknownHostException  
✓ public Socket(InetAddress host, int port, InetAddress interface, int localPort) throws  
IOException
```

✓ **public Socket(String host, int port) throws UnknownHostException, IOException**

```
try {  
    Socket s = new Socket("sgu.edu.vn", 443);  
    if(s.isConnected())  
        System.out.println("Connected");  
} catch (UnknownHostException e) {  
    System.err.println(e);  
} catch (IOException e) {  
    System.err.println(e);  
}
```

Phương thức khác:

- **public InetAddress getInetAddress()**: trả về địa chỉ mà socket kết nối đến
- **public int getPort()**: port number trên máy trạm từ xa đang kết nối với socket.
- **public int getLocalPort()**: port number trên máy cục bộ
- **public InputStream getInputStream() throws IOException**: trả về luồng nhập của socket là đối tượng InputStream cho việc đọc byte từ socket này
- **public OutputStream getOutputStream() throws IOException**: trả về luồng xuất của socket là đối tượng OutputStream.
- **public void close() throws IOException**: Đóng socket

VD:

```
try {  
    Socket s = new Socket("sgu.edu.vn", 443);  
    System.out.println("Connected to " + s.getInetAddress() +  
    " on port " + s.getPort() + " from port " + s.getLocalPort() +  
    " of " + s.getLocalAddress());  
    s.close();  
} catch (UnknownHostException e) {  
    System.err.println(e);  
} catch (IOException e) {  
    System.err.println(e);  
}
```

Connected to sgu.edu.vn/124.158.13.4 on port 443 from port 58619 of /192.168.43.114

- ❑ Lập trình với TCP: chương trình phía client
 - Tạo đối tượng socket và thiết lập kết nối đến server (phải chỉ rõ tham số server)
 - Khai báo luồng nhập/xuất (kiểu byte hoặc char)
 - Truyền thông qua luồng nhập/xuất đã khai báo
 - Đóng socket và giải phóng tài nguyên khi không cần.

❑ Lưu ý:

- Chương trình phía server luôn chạy trước client
- Chương trình server có thể phục vụ nhiều client đồng thời hoặc lặp.

4.2. Chương trình phía Server:

Lớp ServerSocket

Cho phép tạo đối tượng socket phía server và truyền thông với giao thức TCP.

Đối tượng sau khi tại được đặt ở trạng thái lắng nghe (thụ động) chờ tín hiệu kết nối gửi từ client.

Constructor:

- ✓ **public ServerSocket(int port) throws BindException, IOException**
 - Tạo ra đối tượng ServerSocket với số cổng xác định được chỉ ra bởi tham số port.
 - Port=0: cho phép sử dụng một số cổng cho phép nào đó (anonymous port).
 - Trả về ngoại lệ khi socket không thể tạo ra được.
 - Cho phép đáp ứng cực đại tới 50 kết nối đồng thời.

- ✓ **public ServerSocket(int port, int queueLength) throws IOException, BindException**
 - Chỉ ra số kết nối cực đại mà socket có thể đáp ứng đồng thời bởi tham số queueLength

- ✓ **public ServerSocket() throws IOException**
 - Tạo đối tượng ServerSocket nhưng không gắn kết thực sự socket với một port cụ thể.
 - Không thể chấp nhận bất cứ kết nối nào gửi tới nếu chưa bind().

```
ServerSocket ss = new ServerSocket();
SocketAddress http = new InetSocketAddress(80);
ss.bind(http);
```

Phương thức:

Phương thức accept()

- ✓ **public Socket accept() throws IOException**
- ✓ Đặt đối tượng ServerSocket ở trạng thái “nghe” tại số cổng xác định chờ tín hiệu kết nối gửi đến từ client.
- ✓ Khi có tín hiệu kết nối gửi tới phương thức sẽ trả về đối tượng Socket mới để phục vụ kết nối đó.
- ✓ Khi xảy ra lỗi nhập/xuất, phương thức sẽ ném trả về ngoại lệ IOException

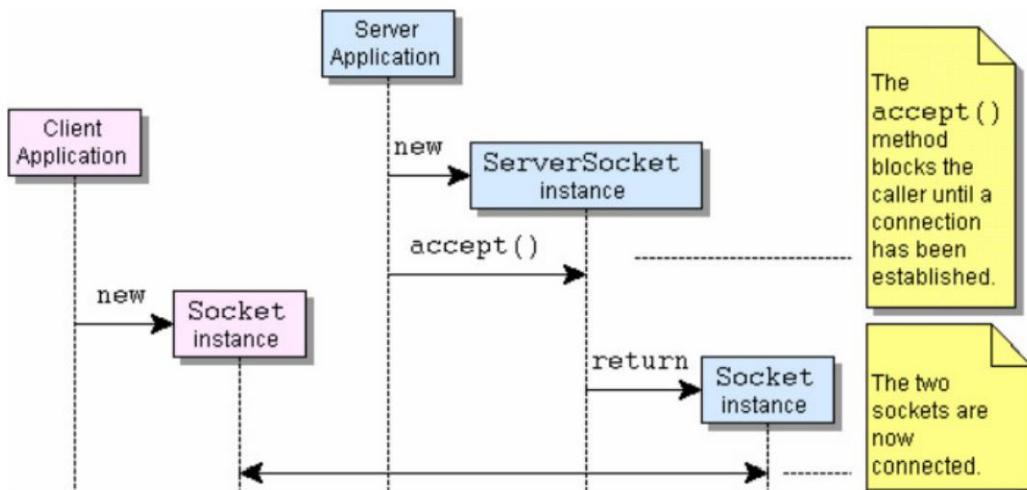
```

ServerSocket server = new ServerSocket(5678);
while (true) {
    Socket conn = server.accept();
    OutputStreamWriter out = new OutputStreamWriter(conn.getOutputStream());
    out.write("You've connected to this server. Bye-bye now.\r\n");
    conn.close();
}

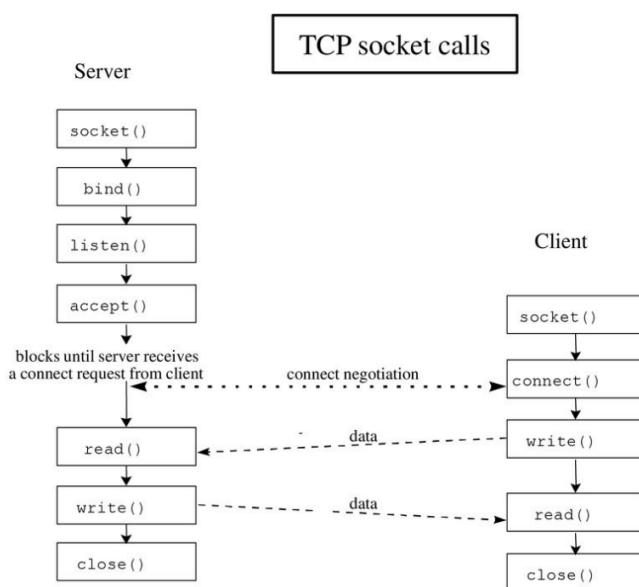
```

- Phương thức close()
 - ✓ **public void close() throws IOException**
 - ✓ Đóng socket và giải phóng tài nguyên.

❑ Lập trình với TCP:



❑ Lập trình với TCP:



❑ Lập trình với TCP: chương trình phía server

- Tạo đối tượng server socket và gán port number
- Lắng nghe yêu cầu kết nối bằng phương thức accept(), với mỗi yêu cầu được chấp thuận, tạo ra đối tượng socket mới để phục vụ:
 - ✓ Lấy InputStream và OutputStream gắn với socket kênh ảo vừa hình thành.
 - ✓ Lặp lại công việc sau:
 - Chờ nhận các yêu cầu
 - Phân tích & thực hiện yêu cầu
 - Tạo thông điệp trả lời
 - Gửi thông điệp trả lời → client
 - Không còn yêu cầu hoặc client kết thúc → đóng socket và quay lại lắng nghe yêu cầu kết nối

VD:

❑ Ví dụ minh họa: chương trình gửi dữ liệu qua lại giữa client và server.

- Client gửi một chuỗi dữ liệu → server
- Server nhận và in chuỗi ra màn hình
- Chương trình lặp lại liên tục cho đến khi client gửi Over thì server đóng socket

❑ Code chương trình server

```
1 // A Java program for a Server
2 import java.net.*;
3 import java.io.*;
4
5 public class Server
6 {
7     private Socket      socket    = null;
8     private ServerSocket server   = null;
9     BufferedWriter out      = null;
10    BufferedReader in       = null;
11
12    public Server(int port)
13    {
14        try
15        {
16            server = new ServerSocket(port);
17            System.out.println("Server started");
18            System.out.println("Waiting for a client ...");
19            socket = server.accept();
20            System.out.println("Client accepted");
21            out = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()));
22            in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
23            String line = "";
```

```

24     while (!line.equals("Over"))
25     {
26         try
27         {
28             line = in.readLine();
29             System.out.println("Server received: " + line);
30         }
31         catch(IOException i)
32         {
33             System.err.println(i);
34         }
35     }
36     System.out.println("Closing connection");
37
38     in.close();
39     out.close();
40     socket.close();
41     server.close();
42 }
43 catch(IOException i)
44 {
45     System.out.println(i);
46 }
47 }
```

27

□ Phần code phía client:

```

1 // A Java program for a Client
2 import java.net.*;
3 import java.io.*;
4
5 public class Client
6 {
7     private Socket socket          = null;
8     BufferedWriter out = null;
9     BufferedReader in = null;
10    BufferedReader stdIn = null;
11
12    public Client(String address, int port) throws UnknownHostException, IOException
13    {
14
15        socket = new Socket(address, port);
16        System.out.println("Connected");
17        out = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()));
18        in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
19        stdIn = new BufferedReader(new InputStreamReader(System.in));
20
21        String line = "";
```

```

22     while (!line.equals("Over"))
23     {
24
25         line = stdIn.readLine();
26         System.out.println("Client sent: " + line);
27         out.write(line);
28         out.newLine();
29         out.flush();
30     }
31
32     in.close();
33     out.close();
34     socket.close();
35
36 }
37
38
39 public static void main(String args[]) throws UnknownHostException, IOException
40 {
41     Client client = new Client("127.0.0.1", 5000);
42 }
43 }
```

▣ Lưu ý khi chạy chương trình:

- Server cần khởi động trước
- Nên phân chia công việc → method, không nên gom trong Main hay constructor
- Lỗi Address already in use xảy ra do port muốn sử dụng bị chiếm, cần kill tiến trình đang chiếm port hoặc sử dụng port khác.
- Lệnh netstat cho biết thông tin về kết nối trên máy tính

C:\Users\Giang Nguyen>netstat -ano findstr 5000				
TCP	0.0.0.0:5000	0.0.0.0:0	LISTENING	17912
TCP	127.0.0.1:50000	0.0.0.0:0	LISTENING	19252
TCP	127.0.0.1:50000	127.0.0.1:59483	ESTABLISHED	19252
TCP	127.0.0.1:59483	127.0.0.1:50000	ESTABLISHED	16656
TCP	[::]:5000	[::]:0	LISTENING	17912

5. Lập trình UPDSocket

5.1. Lớp DatagramPacket:

Tạo gói tin truyền thông với giao thức UDP.

Kế thừa trực tiếp từ lớp Object: **public final class DatagramPacket extends Object**

Gói tin là đối tượng của lớp này chứa 4 thành phần: địa chỉ, dữ liệu truyền, kích thước của gói tin và port number.

Constructor cho gói tin gửi và nhận **khác nhau**:

- ✓ DatagramPacket(byte[] buf, int length)
- ✓ DatagramPacket(byte[] buf, int length, InetAddress address, int port)

- Constructor tạo gói tin (datagram) **nhận** từ mạng:

```
public DatagramPacket(byte[] inBuffer, int length)
```

- ✓ inBuffer: bộ đệm nhập, chứa dữ liệu gói tin nhận
- ✓ length: kích cỡ dữ liệu gói tin nhận, xác định bằng inBuffer.length

```
byte[] inBuffer = new byte[512];
DatagramPacket packet = new DatagramPacket(inBuffer, inBuffer.length);
```

- Constructor tạo gói tin (datagram) **gửi**:

```
public DatagramPacket(byte[] outBuffer, int length, InetAddress destination, int port)
```

- ✓ outBuffer: bộ đệm xuất chứa dữ liệu gửi
- ✓ length: kích cỡ dữ liệu gói tin gửi, tính bằng byte, xác định bằng outBuffer.length
- ✓ dest: địa chỉ nhận gói tin
- ✓ port: cổng nhận gói tin tại đích (dest)

```
String s = "Hello SGU";
byte[] outBuffer = s.getBytes();
InetAddress dest = InetAddress.getByName("localhost");
int port = 1234;
DatagramPacket packet = new DatagramPacket(outBuffer, outBuffer.length, dest, port);
```

- Các phương thức đối với gói tin nhận:

- ✓ *public InetAddress getAddress()*: trả về đối tượng InetAddress của máy gửi trong gói tin nhận.
- ✓ *public int getPort()*: trả về port number của máy gửi trong gói tin nhận.
- ✓ *public byte[] getData()*: trả về dữ liệu chứa trong gói tin dạng byte
- ✓ *public int getLength()*: trả về kích cỡ dữ liệu trong gói tin, tính theo byte.

- Các phương thức đối với gói tin gửi: 4 phương thức bắt đầu bằng set....

5.2. Lớp DatagramSocket:

- Tạo socket truyền thông theo giao thức UDP, gửi nhận gói tin DatagramPacket

- Sử dụng trên cả client và server, không phân chia rõ như TCPSocket

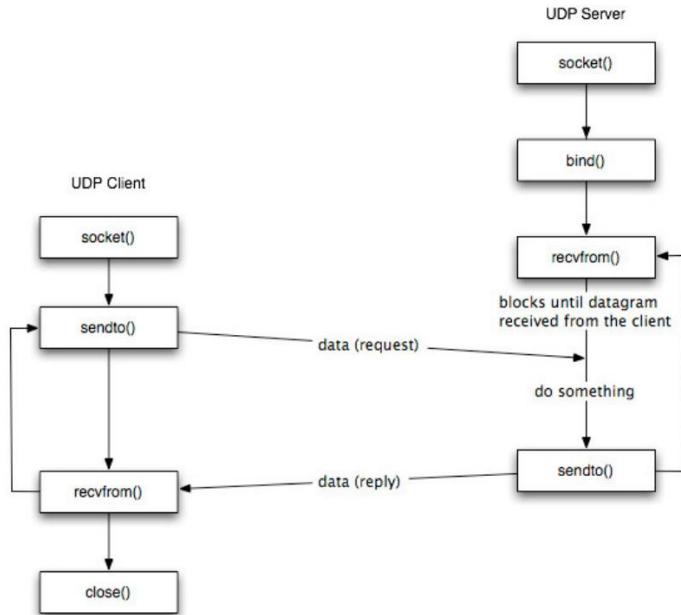
- Constructor:

- ✓ *public DatagramSocket () throws SocketException*: tạo ra UDPSocket sử dụng port number ngẫu nhiên.
- ✓ *public DatagramSocket(int port) throws SocketException*: tạo UDPSocket với port number xác định, thường dùng với server trong mô hình client/server.

- Các phương thức chính:

- ✓ *public void send(DatagramPacket dp) throws IOException*: gửi gói UDP
- ✓ *public void receive(DatagramPacket dp) throws IOException*: nhận gói UDP
- ✓ *public void setSoTimeout(int timeout) throws SocketTimeoutException*
- ✓ *public void close()*

5.3. Xây dựng ứng dụng UDPSocket



❑ Server/Client:

- Tạo đối tượng DatagramSocket kèm port number
- Tạo buffer in/out kiểu byte
- Tạo gói tin DatagramPacket để gửi/nhận dữ liệu.
- Gửi/nhận gói tin bằng phương thức `receive()`/`send()`
- Quay lại bước trên hoặc đóng socket, giải phóng tài nguyên.

9

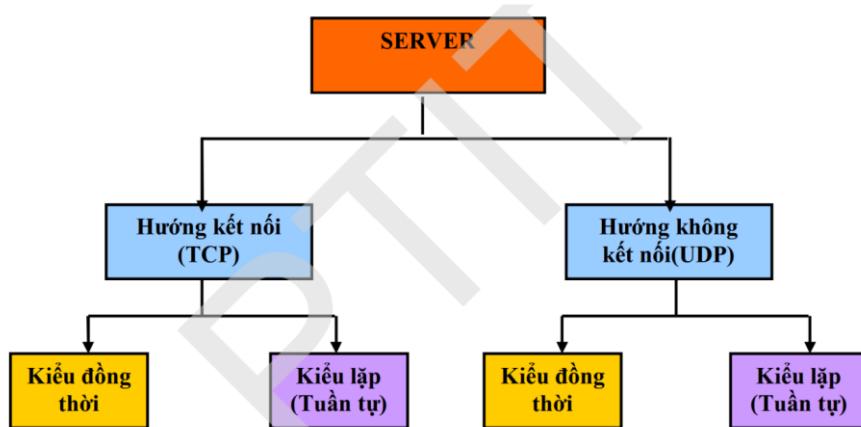
❑ Lưu ý:

- Server phải chạy trước client và trong trạng thái lắng nghe kết nối.
- Client phải gửi DatagramPacket đến server trước
- Server lấy thông tin client trong DatagramPacket để phản hồi

III. KỸ THUẬT XÂY DỰNG ỨNG DỤNG PHÍA SERVER

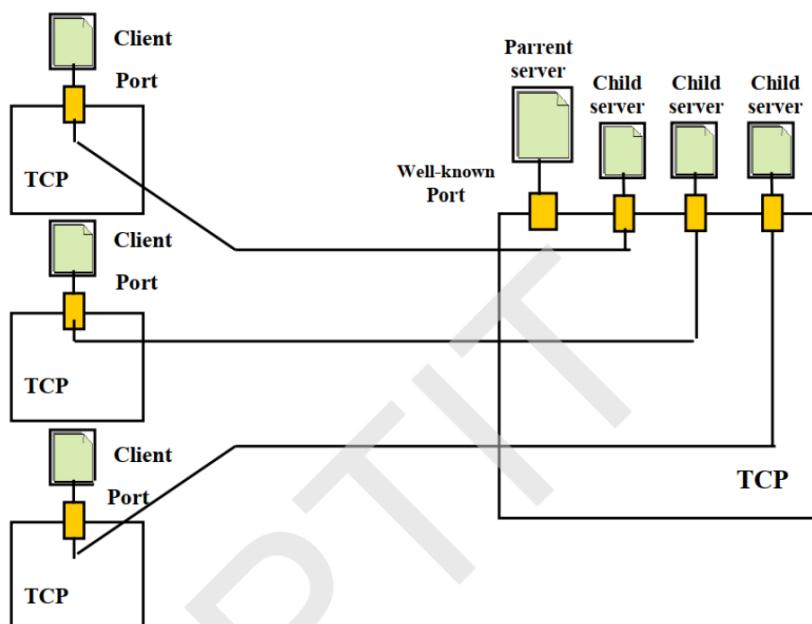
1. Giới thiệu

- ❑ Với mô hình client-server, 1 server có thể phục vụ 1 hoặc nhiều client theo kiểu đồng thời (concurrent) hoặc lặp (iterative)
 - TCPSocket:
 - ✓ Iterative server
 - ✓ Concurrent server: phổ biến
 - UDPSocket
 - ✓ Iterative server: phổ biến
 - ✓ Concurrent server



2. Concurrent TCP Server:

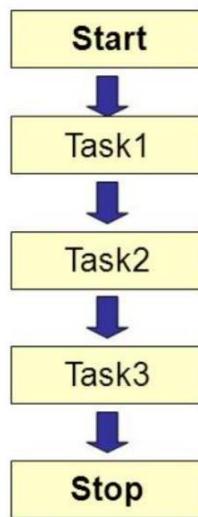
- ❑ Mở 1 cổng quy ước khi khởi tạo để lắng nghe tín hiệu từ client.
- ❑ Tạo server con và mở cổng phụ để kết nối đến mỗi client.
- ❑ Server chính tiếp tục lắng nghe tại cổng quy ước



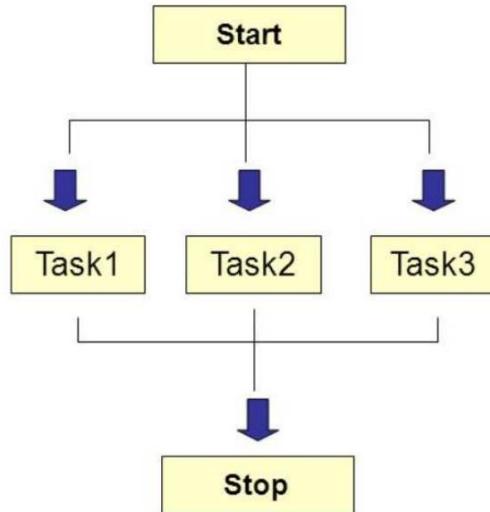
❑ Hai kỹ thuật:

- Chương trình đa tiến trình (process): sinh ra tiến trình con khi có client kết nối → context switch?
- Chương trình đa luồng (tiểu trình, thread): sử dụng hiệu quả tài nguyên, đặc biệt là CPU.

A typical program



Multi-Thread



❑ Thread (luồng): một chuỗi các lệnh được lập trình nhỏ nhất để có thể được quản lý độc lập bởi một bộ định thời, thường là một phần của hệ điều hành.

❑ Ưu điểm của đa luồng (multithread)?

❑ Các vấn đề lưu ý khi sử dụng multithread: synchronized, deadlock

❑ Multithread program gồm:

- Main thread: khởi chạy đầu tiên, sinh ra các thread con, kết thúc sau cùng
- Child thread

❑ Tạo thread trong Java:

- Extends Thread class
- Implement Runnable interface

3. Tạo

Thread:

❑ Extends Thread class:

```

3 class Student extends Thread {
4     String fullName;
5     int age;
6     public Student(String fN, int age) {
7         this.fullName = fN; this.age = age;
8     }
9     public void run() {
10        for(int i = 0 ; i<age; i++)
11            System.out.println("My name is " + fullName + " - i=" + i + "/" + age);
12    }
13 }
14
15 public class Classroom {
16     public static void main(String[] args) {
17         Student s1 = new Student("Anh",25);
18         Student s2 = new Student("Bao",18);
19         Student s3 = new Student("Can",30);
20         s1.start();
21         s2.start();
22         s3.start();
23     }
24 }
  
```

```

My name is Anh - i=0/25
My name is Can - i=0/30
My name is Can - i=1/30
My name is Bao - i=0/18
My name is Can - i=2/30
My name is Can - i=3/30
My name is Can - i=4/30
My name is Can - i=5/30
My name is Anh - i=1/25
My name is Can - i=6/30
My name is Bao - i=1/18
  
```

1

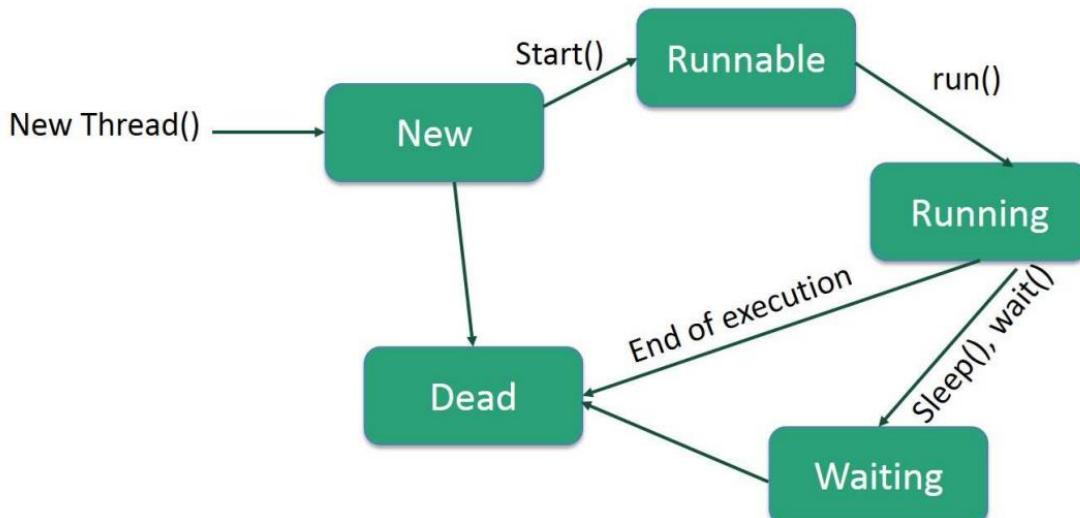
❑ Implements Runnable:

```
3 class Student implements Runnable {  
4     String fullName;  
5     int age;  
6     public Student(String fN, int age) {  
7         this.fullName = fN; this.age = age;  
8     }  
9     public void run() {  
10        for(int i = 0 ; i<=age; i++)  
11            System.out.println("My name is " + fullName + " - i=" + i + "/" + age);  
12    }  
13 }  
14  
15 public class Classroom {  
16     public static void main(String[] args) {  
17         Student s1 = new Student("Anh",25);  
18         Student s2 = new Student("Bao",18);  
19         Thread t1 = new Thread(s1);  
20         Thread t2 = new Thread(s2);  
21         t1.start();  
22         t2.start();  
23     }  
24 }
```

```
My name is Bao - i=0/18  
My name is Bao - i=1/18  
My name is Anh - i=0/25  
My name is Bao - i=2/18  
My name is Anh - i=1/25  
My name is Bao - i=3/18  
My name is Anh - i=2/25  
My name is Bao - i=4/18  
My name is Anh - i=3/25
```

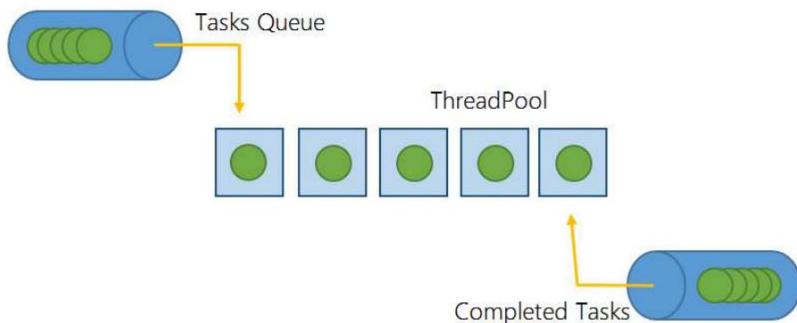
❑ Ưu điểm Implement Runnable interface:

- Không cần kế thừa từ lớp Thread
- ThreadPool

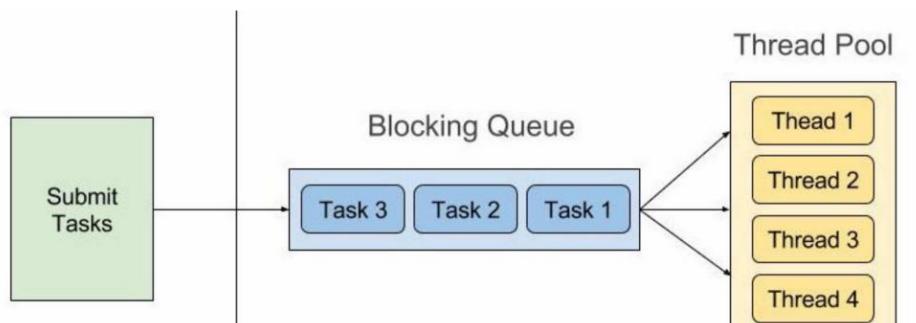


4. ThreadPool:

- ❑ Tạo nhiều thread giải quyết công việc: số lượng thread VS hiệu năng VS cấp phát dữ liệu → Thread pool
- ❑ Thread pool: nhóm các thread đang chờ đợi công việc và tái sử dụng được nhiều lần.
- ❑ Ưu điểm: hiệu năng tốt hơn, tiết kiệm thời gian vì không cần phải tạo thread mới.



- ❑ Nếu request > Thread pool → Blocking queue



- ❑ Java Concurrency API hỗ trợ một vài loại ThreadPool sau:

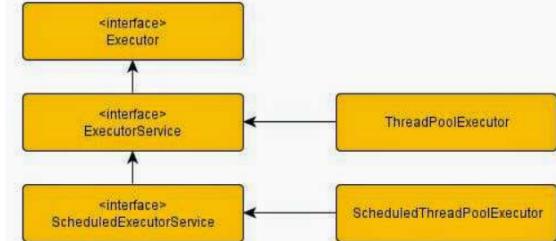
- Cached thread pool: mỗi task sẽ tạo ra thread mới nếu cần, nhưng sẽ tái sử dụng lại các thread cũ
- Fixed thread pool: giới hạn số lượng tối đa của các Thread được tạo ra. Các task khác đến sau phải chờ trong hàng đợi (BlockingQueue)
- Single-threaded pool: chỉ giữ một Thread thực thi một nhiệm vụ một lúc.
- Fork/Join pool: một Thread đặc biệt sử dụng Fork/Join Framework bằng cách tự động chia nhỏ công việc tính toán cho các core xử lý

5. Executor Framework:

Executor là một đối tượng chịu trách nhiệm quản lý các luồng và thực hiện các tác vụ Runnable được yêu cầu xử lý.

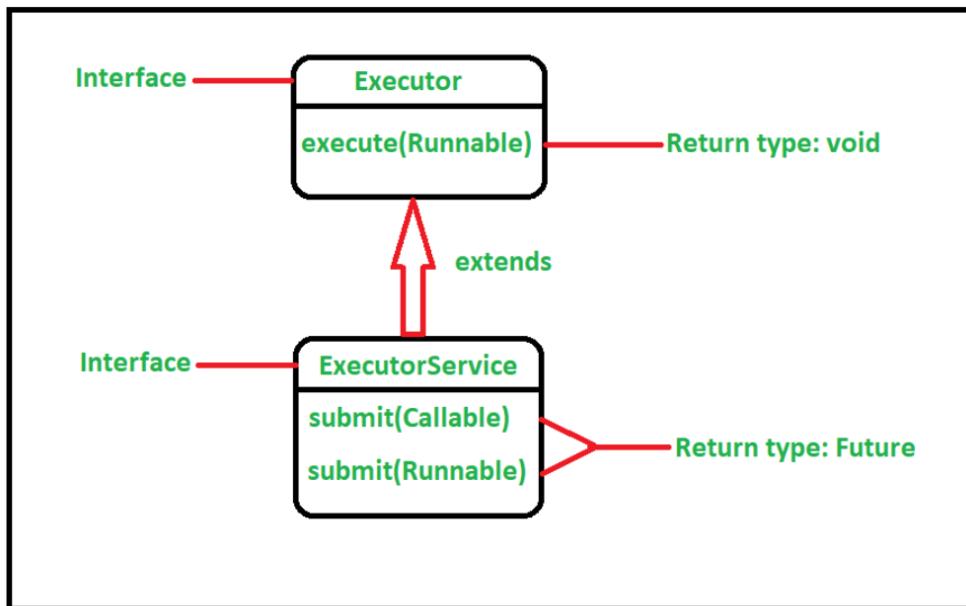
Executor tách riêng các chi tiết của việc tạo Thread, lập lịch (scheduling), ... nhằm tập trung phát triển logic của tác vụ mà không quan tâm đến các chi tiết quản lý Thread.

- “Executor framework” trong gói java.util.concurrent giúp tạo và quản lý các “ThreadPool” và “Thread Factories”.
 - Executor: là interface cha của tất cả các Executor, method: execute(Runnable)
 - ExecutorService: theo dõi tiến trình của các tác vụ trả về giá trị (Callable) thông qua đối tượng Future, và quản lý việc kết thúc các luồng. Method: submit() và shutdown().
 - ScheduledExecutorService: là một ExecutorService có thể lên lịch cho các tác vụ để thực thi sau một khoảng thời gian nhất định, hoặc để thực hiện định kỳ. Method: schedule(), scheduleAtFixedRate() & scheduleWithFixedDelay()



28

❑ ExecutorService



❑ Tao Executor:

- **newSingleThreadExecutor()**: ThreadPool chỉ có 1 thread và task thực thi tuần tự.
 - **newCachedThreadPool()**: ThreadPool có nhiều Thread và các task sẽ được xử lý song song. Các Thread cũ sau khi xử lý xong sẽ được sử dụng lại cho task mới. Mặc định, Thread không được sử dụng trong vòng 60 giây thì sẽ bị tắt.
 - **newFixedThreadPool(int nThreads)**: ThreadPool chứa tối đa nThreads. Khi Pool đạt đến giá trị tối đa nThreads, các Thread còn lại sẽ được đưa vào Blocking Queue.
 - **newScheduledThreadPool(int corePoolSize)**: tương tự như newCachedThreadPool() nhưng sẽ có thời gian delay giữa các Thread.
 - **newSingleThreadScheduledExecutor()**: tương tự như newSingleThreadExecutor() nhưng sẽ có khoảng thời gian delay giữa các Thread.

VD:

□ Worker Thread

```
5 public class Worker implements Runnable {  
6     private String task;  
7     public Worker(String s) {  
8         this.task = s;  
9     }  
10    public void run() {  
11        System.out.println(Thread.currentThread().getName() + " starting. Task = " + task);  
12        try {  
13            Random rand = new Random();  
14            Thread.sleep(rand.nextInt(3000)+500); // sleep between 500 - 3500 ms  
15        } catch (InterruptedException e) {e.printStackTrace();}  
16        System.out.println(Thread.currentThread().getName() + " finished.");  
17    }  
18 }
```

□ newSingleThreadExecutor()

```
3 import java.util.concurrent.ExecutorService;  
4 import java.util.concurrent.Executors;  
5  
6 public class SingleThreadExecutor {  
7     public static void main(String[] args) {  
8         ExecutorService executor =  
9             Executors.newSingleThreadExecutor();  
10        for(int i=1; i<=7; i++) {  
11            Runnable worker =  
12                new Worker(Integer.toString(i));  
13            executor.execute(worker);  
14        }  
15        executor.shutdown();  
16        // Wait until all threads are finish  
17        while(!executor.isTerminated()) {}  
18        System.out.println("All threads finished");  
19    }  
20 }  
21
```

```
pool-1-thread-1 Starting. Task = 1  
pool-1-thread-1 Finished.  
pool-1-thread-1 Starting. Task = 2  
pool-1-thread-1 Finished.  
pool-1-thread-1 Starting. Task = 3  
pool-1-thread-1 Finished.  
pool-1-thread-1 Starting. Task = 4  
pool-1-thread-1 Finished.  
pool-1-thread-1 Starting. Task = 5  
pool-1-thread-1 Finished.  
pool-1-thread-1 Starting. Task = 6  
pool-1-thread-1 Finished.  
pool-1-thread-1 Starting. Task = 7  
pool-1-thread-1 Finished.  
All threads finished
```

32

□ newFixedThreadPool(int nThreads)

```
3 import java.util.concurrent.Executors;  
4 import java.util.concurrent.ExecutorService;  
5  
6 public class FixedThreadPool {  
7     public static void main(String[] args) {  
8         ExecutorService executor =  
9             Executors.newFixedThreadPool(3);  
10        for(int i=1; i<=7; i++) {  
11            Worker worker =  
12                new Worker(Integer.toString(i));  
13            executor.execute(worker);  
14        }  
15        executor.shutdown();  
16        while(!executor.isTerminated()) {}  
17        System.out.println("All threads finished");  
18    }  
19 }
```

```
pool-1-thread-1 Starting. Task = 1  
pool-1-thread-3 Starting. Task = 3  
pool-1-thread-2 Starting. Task = 2  
pool-1-thread-3 Finished.  
pool-1-thread-3 Starting. Task = 4  
pool-1-thread-3 Finished.  
pool-1-thread-3 Starting. Task = 5  
pool-1-thread-1 Finished.  
pool-1-thread-1 Starting. Task = 6  
pool-1-thread-1 Finished.  
pool-1-thread-1 Starting. Task = 7  
pool-1-thread-2 Finished.  
pool-1-thread-3 Finished.  
pool-1-thread-1 Finished.  
All threads finished
```

33

❑ newCachedThreadPool()

```
3 import java.util.concurrent.ExecutorService;
4 import java.util.concurrent.Executors;
5
6 public class CachedThreadPool {
7     public static void main(String[] args) {
8         ExecutorService executor =
9             Executors.newCachedThreadPool();
10    for(int i=1; i<=7; i++) {
11        Worker worker =
12            new Worker(Integer.toString(i));
13        executor.execute(worker);
14    }
15    executor.shutdown();
16    while(!executor.isTerminated()) {}
17    System.out.println("All threads finished");
18 }
19 }
```

pool-1-thread-3 Starting. Task = 3
pool-1-thread-6 Starting. Task = 6
pool-1-thread-5 Starting. Task = 5
pool-1-thread-1 Starting. Task = 1
pool-1-thread-2 Starting. Task = 2
pool-1-thread-4 Starting. Task = 4
pool-1-thread-7 Starting. Task = 7
pool-1-thread-2 Finished.
pool-1-thread-4 Finished.
pool-1-thread-1 Finished.
pool-1-thread-5 Finished.
pool-1-thread-7 Finished.
pool-1-thread-3 Finished.
pool-1-thread-6 Finished.
All threads finished

34

6. Ví dụ ThreadPoolExecutor:

ThreadPool thông thường (ExecutorService) không đủ linh động theo tình huống: số lượng thread cố định hoặc cho phép tạo quá nhiều thread.

ThreadPoolExecutor cho phép tùy biến số lượng Thread theo kịch bản.

- + corePoolSize: số lượng Thread mặc định trong Pool
- + maxPoolSize: số lượng tối đa Thread trong Pool
- + queueCapacity: số lượng tối đa của BlockingQueue

❑ Ví dụ ThreadPoolExecutor:

- corePoolSize: 5
- maxPoolSize: 15
- queueCapacity: 100

❑ Giải thích:

- Khi có request, ThreadPoolExecutor sẽ tạo trong Pool tối đa 5 thread (**corePoolSize**).
- Khi số lượng thread vượt quá 5. ThreadPoolExecutor sẽ cho thread mới vào hàng đợi.
- Khi số lượng hàng đợi full 100 (**queueCapacity**) → bắt đầu tạo thêm Thread mới.
- Số Thread mới được tạo tối đa là 15 (**maxPoolSize**).
- Khi Request vượt quá số lượng 15 thread. Request sẽ bị từ chối!

- ❑ Ví dụ ThreadPoolExecutor:

```
3 import java.util.concurrent.ArrayBlockingQueue;
4 import java.util.concurrent.ThreadPoolExecutor;
5 import java.util.concurrent.TimeUnit;
6
7 public class ThreadPoolExecutorExample {
8     public static void main(String[] args) {
9         int corePoolSize = 1;
10        int maximumPoolSize = 3;
11        int queueCapacity = 8;
12        ThreadPoolExecutor executor =
13            new ThreadPoolExecutor(corePoolSize, maximumPoolSize, 10,
14                TimeUnit.SECONDS, new ArrayBlockingQueue<>(queueCapacity));
15        for (int i = 1; i <= 10; i++) {
16            executor.execute(new Worker("Request-" + i));
17        }
18        executor.shutdown();
19        while (!executor.isTerminated()) {}
20        System.out.println("All threads finished");
21    }
22 }
```

Lưu ý khi sử dụng ExecutorService

- ❑ Phương thức `shutdown()`: ExecutorService sẽ từ chối nhận thêm các task, tất cả các task được thêm vào trước khi gọi `shutdown()` đều sẽ được thực thi, các task thêm sau sẽ bị từ chối (`rejected`).
- ❑ Phương thức `shutdownNow()`: tắt ExecutorService ngay lập tức, các task trong Queue cũng sẽ bị loại bỏ.

7. Ví dụ TCPSocket multithread

Ví dụ TCPSocket multithread client -> server, Server code

```
9 public class Server {  
10    public static int port = 1234;  
11    public static int numThread = 2;  
12    private static ServerSocket server = null;  
13  
14    public static void main(String[] args) throws IOException {  
15        ExecutorService executor = Executors.newFixedThreadPool(numThread);  
16        try {  
17            server = new ServerSocket(port);  
18            System.out.println("Server binding at port " + port);  
19            System.out.println("Waiting for client...");  
20            while(true) {  
21                Socket socket = server.accept();  
22                executor.execute(new Worker(socket));  
23            }  
24        } catch (IOException e) {  
25            System.out.println(e);  
26        } finally {  
27            if(server!=null)  
28                server.close();  
29        }  
30    }  
31 }
```

Ví dụ TCPSocket multithread client -> server, Client code

```
15    public static void main(String[] args) throws IOException {  
16        try {  
17            socket = new Socket(host, port);  
18            System.out.println("Client connected");  
19            BufferedWriter out = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()));  
20            BufferedReader stdIn = new BufferedReader(new InputStreamReader(System.in));  
21            String input;  
22            while(true) {  
23                System.out.print("Client sent: ");  
24                input = stdIn.readLine();  
25                out.write(input + '\n');  
26                out.flush();  
27                if(input.equals("bye"))  
28                    break;  
29            }  
30        } catch (IOException e) {  
31            System.out.println(e);  
32        } finally {  
33            if(socket!=null) {  
34                socket.close();  
35                System.out.println("Client socket closed");  
36            }  
37        }  
38    }
```

BÀI TẬP

1. ChatRoom TCP:

* *Client*

```
1 package ChatRoomTCP;
2
3 import java.io.DataInputStream;
4 import java.io.DataOutputStream;
5 import java.io.IOException;
6 import java.net.InetAddress;
7 import java.net.Socket;
8 import java.util.Scanner;
9
10 public class Client {
11     private InetAddress host;
12     private int port;
13
14     public Client(InetAddress host, int port) {
15         this.host = host;
16         this.port = port;
17     }
18
19     private void execute() throws IOException {
20         //Phần bổ sung
21         Scanner sc = new Scanner(System.in);
22         System.out.print("Nhập vào tên của bạn: ");
23         String name = sc.nextLine();
24
25         Socket client = new Socket(host, port);
26         ReadClient read = new ReadClient(client);
27         read.start();
28         WriteClient write = new WriteClient(client, name);
29         write.start();
30     }
31
32
33     public static void main(String[] args) throws IOException {
34         Client client = new Client(InetAddress.getLocalHost(), 15797);
35         client.execute();
36     }
37 }
38
39 class ReadClient extends Thread{
40     private Socket client;
41
42     public ReadClient(Socket client) {
43         this.client = client;
44     }
45 }
```

```

46     @Override
47     public void run() {
48         DataInputStream dis = null;
49         try {
50             dis = new DataInputStream(client.getInputStream());
51             while(true) {
52                 String sms = dis.readUTF();
53                 System.out.println(sms);
54             }
55         } catch (Exception e) {
56             try {
57                 dis.close();
58                 client.close();
59             } catch (IOException ex) {
60                 System.out.println("Ngắt kết nối Server");
61             }
62         }
63     }
64 }
65
66 class WriteClient extends Thread{
67     private Socket client;
68     private String name;
69
70     public WriteClient(Socket client, String name) {
71         this.client = client;
72         this.name = name;
73     }
74
75     @Override
76     public void run() {
77         DataOutputStream dos = null;
78         Scanner sc = null;
79         try {
80             dos = new DataOutputStream(client.getOutputStream());
81             sc = new Scanner(System.in);
82             while(true) {
83                 String sms = sc.nextLine();
84                 dos.writeUTF(name + ":" + sms);
85             }
86         } catch (Exception e) {
87             try {
88                 dos.close();
89                 client.close();
90             } catch (IOException ex) {
91                 System.out.println("Ngắt kết nối Server");
92             }
93         }
94     }
95 }
96 }
```

* Server

```

1 package ChatRoomTCP;
2
3 import java.io.DataInputStream;
4 import java.io.DataOutputStream;
5 import java.io.IOException;
6 import java.net.ServerSocket;
7 import java.net.Socket;
8 import java.util.ArrayList;
9 import java.util.Scanner;
10
11 public class Server {
12     private int port;
13     public static ArrayList<Socket> listSK;
14
15     public Server(int port) {
16         this.port = port;
17     }
18
19     private void execute() throws IOException {
20         ServerSocket server = new ServerSocket(port);
21         WriteServer write = new WriteServer();
22         write.start();
23         System.out.println("Server is listening...");
24         while (true) {
25             Socket socket = server.accept();
26             System.out.println("Đã kết nối với " + socket);
27             Server.listSK.add(socket);
28             ReadServer read = new ReadServer(socket);
29             read.start();
30         }
31     }
32
33     public static void main(String[] args) throws IOException {
34         Server.listSK = new ArrayList<>();
35         Server server = new Server(15797);
36         server.execute();
37     }
38
39 }
40
41 class ReadServer extends Thread {
42     private Socket socket;
43
44     public ReadServer(Socket socket) {
45         this.socket = socket;
46     }
47
48     @Override

```

```

48     @Override
49     public void run() {
50         try {
51             DataInputStream dis = new DataInputStream(socket.getInputStream());
52             while (true) {
53                 String sms = dis.readUTF();
54                 if(sms.contains("exit")) {
55                     Server.listSK.remove(socket);
56                     System.out.println("Đã ngắt kết nối với " + socket);
57                     dis.close();
58                     socket.close();
59                     continue; //Ngắt kết nối rồi
60                 }
61                 for (Socket item : Server.listSK) {
62                     if(item.getPort() != socket.getPort()) {
63                         DataOutputStream dos = new DataOutputStream(item.getOutputStream());
64                         dos.writeUTF(sms);
65                     }
66                 }
67                 System.out.println(sms);
68             }
69         } catch (Exception e) {
70             try {
71                 socket.close();
72             } catch (IOException ex) {
73                 System.out.println("Ngắt kết nối Server");
74             }
75         }
76     }
77 }
78
79 class WriteServer extends Thread {
80
81     @Override
82     public void run() {
83         DataOutputStream dos = null;
84         Scanner sc = new Scanner(System.in);
85         while (true) {
86             String sms = sc.nextLine();      //Đang đợi Server nhập dữ liệu
87             try {
88                 for (Socket item : Server.listSK) {
89                     dos = new DataOutputStream(item.getOutputStream());
90                     dos.writeUTF("Server: " + sms);
91                 }
92             } catch (IOException e) {
93                 e.printStackTrace();
94             }
95         }
96     }
97 }
98 }
```

2. Giải phương trình bậc 1

* *Client*

```
1 package Client;
2
3 import java.io.DataInputStream;
4 import java.io.DataOutputStream;
5 import java.io.IOException;
6 import java.net.InetAddress;
7 import java.net.Socket;
8 import java.util.Scanner;
9
10 public class Client {
11     private InetAddress host;
12     private int port;
13     private Scanner sc = new Scanner(System.in);
14
15     public Client(InetAddress host, int port) {
16         this.host = host;
17         this.port = port;
18     }
19
20     private void execute() throws IOException {
21         //Cấu hình Client
22         Socket client = new Socket(host, port);
23         //Nhập dữ liệu
24         int a = input("Nhập vào a: ");
25         int b = input("Nhập vào b: ");
26
27         DataInputStream dis = new DataInputStream(client.getInputStream());
28         DataOutputStream dos = new DataOutputStream(client.getOutputStream());
29
30         //Gửi dữ liệu đến Server
31         dos.writeInt(a);
32         dos.writeInt(b);
33
34         //Nhận dữ liệu từ Server
35         System.out.println(dis.readUTF());
36
37         dis.close();
38         dos.close();
39         client.close();
40     }
41
42     public static void main(String[] args) throws IOException {
43         Client client = new Client(InetAddress.getLocalHost(), 15797);
44         client.execute();
45     }
46
47     private int input(String request) {
48         int number = 0;
49         boolean flag = true;
50         do {
51             try {
52                 System.out.print(request);
53                 number = Integer.parseInt(sc.nextLine());
54                 flag = false;
55             } catch (Exception e) {
56                 System.out.println("Dữ liệu không đúng định dạng. Vui lòng nhập lại: ");
57             }
58         } while (flag);
59         return number;
60     }
61 }
```

```

3 import java.io.DataInputStream;
4 import java.io.DataOutputStream;
5 import java.io.IOException;
6 import java.net.ServerSocket;
7 import java.net.Socket;
8
9 public class Server {
10     private int port;
11
12     public Server(int port) {
13         this.port = port;
14     }
15
16     private void execute() throws IOException {
17         ServerSocket server = new ServerSocket(port);
18         System.out.println("Server is listening...");
19
20         while(true) {
21             Socket socket = server.accept();
22             PTBac1 ptBac1 = new PTBac1(socket);
23             ptBac1.start();
24         }
25     }
26
27     public static void main(String[] args) throws IOException {
28         Server server = new Server(15797);
29         server.execute();
30     }
31 }
32
33 class PTBac1 extends Thread{
34     private Socket socket;
35
36     public PTBac1(Socket socket) {
37         this.socket = socket;
38     }
39
40     @Override
41     public void run() {
42         try {
43             DataInputStream dis = new DataInputStream(socket.getInputStream());
44             DataOutputStream dos = new DataOutputStream(socket.getOutputStream());
45
46             int a = dis.readInt();
47             int b = dis.readInt();
48             if(a == 0) {
49                 if(b == 0) {
50                     dos.writeUTF("Phương trình vô số nghiệm");
51                 }else {
52                     dos.writeUTF("Phương trình vô nghiệm");
53                 }
54             }else {
55                 double result = (double)-b/a;
56                 dos.writeUTF("Phương trình có nghiệm = " + result);
57             }
58             dis.close();
59             dos.close();
60         } catch (Exception e) {
61             e.printStackTrace();
62         }
63     }

```

3. Giải phương trình bậc 2 – UPD

* *Client*

```

1 package GiaiPTBac2_UDP;
2
3 import java.io.IOException;
4 import java.net.DatagramPacket;
5 import java.net.DatagramSocket;
6 import java.net.InetAddress;
7 import java.util.Scanner;
8
9
10 public class Client {
11     private InetAddress host;
12     private int port;
13     private Scanner sc = new Scanner(System.in);
14
15     public Client(InetAddress host, int port) {
16         this.host = host;
17         this.port = port;
18     }
19
20     private void execute() throws IOException {
21         //Cấu hình Client
22         DatagramSocket client = new DatagramSocket();
23
24         //Nhập dữ liệu
25         int a = input("Nhập vào a: ");
26         int b = input("Nhập vào b: ");
27         int c = input("Nhập vào c: ");
28
29         //Đóng gói thành Packet và gửi đến Server
30         DatagramPacket a_DP = createPacket(a, 1);
31         DatagramPacket b_DP = createPacket(b, 2);
32         DatagramPacket c_DP = createPacket(c, 3);
33
34         client.send(a_DP);
35         client.send(b_DP);
36         client.send(c_DP);
37
38         //Nhận dữ liệu từ Server
39         String [] result = receiveData(client).split("_");
40         double value = Double.parseDouble(result[0]);
41         int key      = Integer.parseInt(result[1]);
42         if(key == -1) {
43             System.out.println("Phương trình vô nghiệm");
44         }else if(key == -2) {
45             System.out.println("Phương trình vô số nghiệm");
46         }else if(key == -3) {
47             System.out.println("Phương trình có 1 nghiệm = " + value);
48         }else if(key == -4) {
49             String [] result1 = receiveData(client).split("_");
50             double value_1 = Double.parseDouble(result1[0]);
51             System.out.println("Phương trình có 2 nghiệm phân biệt: ");
52             System.out.println("x1 = " + value);
53             System.out.println("x2 = " + value_1);
54         }
55         client.close();
56     }
57 }
```

```

58     public static void main(String[] args) throws IOException {
59         Client client = new Client(InetAddress.getLocalHost(), 1597);
60         client.execute();
61     }
62
63     private String receiveData(DatagramSocket server) throws IOException {
64         byte[] temp = new byte[1024];
65         DatagramPacket recieve_Packet = new DatagramPacket(temp, temp.length);
66         server.receive(recieve_Packet);
67         return new String(recieve_Packet.getData()).trim();
68     }
69
70     private DatagramPacket createPacket(int value, int index) {
71         String str = String.valueOf(value) + "_" + index;
72         byte[] arrData = str.getBytes();
73         return new DatagramPacket(arrData, arrData.length, host, port);
74     }
75
76     private int input(String request) {
77         int number = 0;
78         boolean flag = true;
79         do {
80             try {
81                 System.out.print(request);
82                 number = Integer.parseInt(sc.nextLine());
83                 flag = false;
84             } catch (Exception e) {
85                 System.out.println("Dữ liệu không đúng định dạng. Vui lòng nhập lại: ");
86             }
87         } while (flag);
88         return number;
89     }
90
91
92 }

```

* Server

```

1 package GiaiPTBac2_UDP;
2
3 import java.io.IOException;
4 import java.net.DatagramPacket;
5 import java.net.DatagramSocket;
6 import java.net.InetAddress;
7 import java.net.SocketException;
8 import java.util.ArrayList;
9
10 public class Server {
11     private int port;
12     private InetAddress clientIP;
13     private int clientPort;
14     public static ArrayList<Data> listSK;
15
16     public Server(int port) {
17         this.port = port;
18     }
19

```

```

20     private void execute() throws IOException {
21         // Tạo Server Socket
22         DatagramSocket server = new DatagramSocket(port);
23         System.out.println("Server is listening....");
24         while (true) {
25             String[] arrData = receiveData(server).split("_");
26             int key = Integer.parseInt(arrData[1]);
27             int value = Integer.parseInt(arrData[0]);
28             if (checkClientExist(clientIP, clientPort) == false) { // Client này kết nối lần đầu
29                 Data data = new Data(clientIP, clientPort);
30                 // Gán dữ liệu vào
31                 if (key == 1) {
32                     data.setA(value);
33                 } else if (key == 2) {
34                     data.setB(value);
35                 } else if (key == 3) {
36                     data.setC(value);
37                 }
38                 listSK.add(data);
39             } else { // Đã tồn tại
40                 for (Data item : listSK) {
41                     if (item.getHost().equals(clientIP) && item.getPort() == clientPort) {
42                         // Gán dữ liệu vào
43                         if (key == 1) {
44                             item.setA(value);
45                         } else if (key == 2) {
46                             item.setB(value);
47                         } else if (key == 3) {
48                             item.setC(value);
49                         }
50                     }
51                 }
52                 // Kiểm tra xem Client này đã đưa đủ dữ liệu thì giải PT
53                 if(item.getCount() == 3) {
54                     giaoPTBac2(server, item);
55                     // Sau khi tính toán cũng xóa Client trong danh sách ra
56                     listSK.remove(item);
57                     break;
58                 }
59             }
60         }
61     }
62 }
63
64     public static void main(String[] args) throws IOException {
65         Server.listSK = new ArrayList<>();
66         Server server = new Server(15797);
67         server.execute();
68     }
69
70     private void giaoPTBac2(DatagramSocket server, Data data) throws IOException {
71         /*
72          * -1: vô nghiệm      -2: vô số nghiệm      -3: 1 nghiệm kép      -4: 2 nghiệm
73          * */
74         int a = data.getA();
75         int b = data.getB();
76         int c = data.getC();
77
78         // Xử lý và gửi về Client
79         if(a == 0) {

```

```

78         //Xử lý và gửi về Client
79         if(a == 0) {
80             if(b == 0) {
81                 if(c == 0) { //Phương trình vô số nghiệm
82                     sendData(0, -2, server, clientIP, clientPort);
83                 } else { //Phương trình vô nghiệm
84                     sendData(0, -1, server, clientIP, clientPort);
85                 }
86             } else { //Phương trình có 1 nghiệm
87                 sendData((-c/b), -3, server, clientIP, clientPort);
88             }
89             return;
90         }
91
92         double delta = (b*b) - (4*a*c);
93         double x1;
94         double x2;
95         if(delta > 0) { //Phương trình có 2 nghiệm
96             x1 = (double) ((-b + Math.sqrt(delta)) / (2 * a));
97             x2 = (double) ((-b - Math.sqrt(delta)) / (2 * a));
98             sendData(x1, -4, server, clientIP, clientPort);
99             sendData(x2, -4, server, clientIP, clientPort);
100        } else if(delta == 0) { //Phương trình có 1 nghiệm
101            x1 = (-b / (2 * a));
102            sendData(x1, -3, server, clientIP, clientPort);
103        } else { //Phương trình vô nghiệm
104            sendData(0, -1, server, clientIP, clientPort);
105        }
106    }
107
108    private void sendData(double value, int index, DatagramSocket server, InetAddress clientIP, int clientPort) throws IOException {
109        byte[] temp = new byte[1024];
110        String str = String.valueOf(value) + "_" + index;
111        temp = str.getBytes();
112        DatagramPacket send_result_Packet = new DatagramPacket(temp, temp.length, clientIP, clientPort);
113        server.send(send_result_Packet);
114    }
115
116    private String recieveData(DatagramSocket server) throws IOException {
117        byte[] temp = new byte[1024];
118        DatagramPacket recieve_Packet = new DatagramPacket(temp, temp.length);
119        server.receive(recieve_Packet);
120        clientIP = recieve_Packet.getAddress();
121        clientPort = recieve_Packet.getPort();
122        return new String(recieve_Packet.getData()).trim();
123    }
124
125    private boolean checkClientExist(InetAddress clientIP, int clientPort) {
126        for (Data item : listSK) {
127            if (item.getHost().equals(clientIP) && item.getPort() == clientPort) {
128                return true;
129            }
130        }
131        return false;
132    }
133}
134
135}
136
137 class Data {
138     private InetAddress host;
139     private int port;
140     private int a;
141     private int b;
142     private int c;
143     private int count;
144
145     public Data(InetAddress host, int port) {
146         this.host = host;
147         this.port = port;
148         this.a = 0;
149         this.b = 0;
150         this.c = 0;
151         this.count = 0;
152     }
153
public InetAddress getHost() {
    return host;
}

public void setHost(InetAddress host) {
    this.host = host;
}

public void setPort(int port) {
    this.port = port;
}

```

4. Login

```
1  package LoginTCP;
2
3  import java.io.DataInputStream;
4  import java.io.DataOutputStream;
5  import java.io.IOException;
6  import java.net.InetAddress;
7  import java.net.Socket;
8  import java.net.UnknownHostException;
9  import java.util.Scanner;
10
11 public class Client {
12     private InetAddress host;
13     private int port;
14     public Client(InetAddress host, int port) {
15         this.host = host;
16         this.port = port;
17     }
18
19     private void execute() throws IOException {
20         Socket client = new Socket(host, port);
21         DataInputStream dis = new DataInputStream(client.getInputStream());
22         DataOutputStream dos = new DataOutputStream(client.getOutputStream());
23
24         Scanner sc = new Scanner(System.in);
25
26         int id = 0;
27         String pass = "";
28         boolean flag = true;
29         int result = 0;
30         do {
31             System.out.print("Nhập vào id: ");
32             try {
33                 id = Integer.parseInt(sc.nextLine());
34             } catch (NumberFormatException e) {
35                 System.out.println("id phải là số nguyên. Vui lòng nhập lại!");
36                 System.out.print("Nhập vào id: ");
37                 id = Integer.parseInt(sc.nextLine());
38             }
39
40             System.out.print("Nhập vào password: ");
41             pass = sc.nextLine();
42             dos.writeInt(id);
43             dos.writeUTF(pass);
44             result = dis.readInt();
45             if(result == -1) {
46                 System.out.println("Sai password. Vui lòng nhập lại");
47             }else if(result == -2) {
48                 System.out.println("User không tồn tại. Vui lòng nhập lại");
49             }else {
50                 System.out.println("Số tiền của bạn là: " + result);
51                 flag = false;
52             }
53
54         } while (flag);    60         public static void main(String[] args) throws IOException {
55             dis.close();      61             Client client = new Client(InetAddress.getLocalHost(), 15797);
56             dos.close();      62             client.execute();
57             client.close();   63         }
58     }                      64 }
```

* Server:

```

3 import java.io.DataInputStream;
4 import java.io.DataOutputStream;
5 import java.io.IOException;
6 import java.net.ServerSocket;
7 import java.net.Socket;
8 import java.sql.Connection;
9 import java.sql.PreparedStatement;
10 import java.sql.ResultSet;
11 import java.sql.SQLException;
12
13 public class Server {
14     private int port;
15
16     public Server(int port) {
17         this.port = port;
18     }
19
20     private void execute() throws IOException, SQLException {
21         ServerSocket server = new ServerSocket(port);
22         System.out.println("Server is listening....");
23         Socket socket = server.accept();
24         DataInputStream dis = new DataInputStream(socket.getInputStream());
25         DataOutputStream dos = new DataOutputStream(socket.getOutputStream());
26
27         int id = 0;
28         String pass = "";
29         boolean flag = true;
30         int result = 0;
31         do {
32             id = dis.readInt();
33             pass = dis.readUTF();
34             Connection connect = DBConnection.getConnection();
35             String query = "SELECT dbo.FN_Login(?, ?) AS result";
36             PreparedStatement ps = connect.prepareStatement(query);
37             ps.setInt(1, id);
38             ps.setString(2, pass);
39             ResultSet rs = ps.executeQuery();
40             rs.next();
41             result = rs.getInt("result");
42             if(result == -1) {
43                 dos.writeInt(-1);
44             } else if(result == -2) {
45                 dos.writeInt(-2);
46             } else {
47                 dos.writeInt(result);
48                 flag = false;
49             }
50             DBConnection.closeConnection(connect);
51         } while (flag);
52         dis.close();
53         dos.close();
54         socket.close();
55         server.close();
56     }
57     public static void main(String[] args) throws IOException, SQLException {
58         Server server = new Server(15797);
59         server.execute();
60     }
61 }
62
63 1

```

5. Database Connection

```
1 package LoginTCP;
2
3 import java.sql.Connection;
4 import java.sql.DriverManager;
5 import java.sql.SQLException;
6
7 public class DBConnection {
8     public static Connection getConnection(){
9         Connection connection = null;
10        try {
11            Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
12            String url = "jdbc:sqlserver://localhost:1433; databaseName=NhanVien";
13            String user = "sa";
14            String pass = "123456";
15            connection = DriverManager.getConnection(url, user, pass);
16        } catch (Exception ex) {
17            ex.printStackTrace();
18        }
19        return connection;
20    }
21
22    public static void closeConnection(Connection con){
23        if(con != null){
24            try {
25                con.close();
26            } catch (SQLException ex) {
27                ex.printStackTrace();
28            }
29        }
30    }
31 }
```

6. Xử lý số:

* Client:

```
1 package menuTCP;
2
3 import java.io.DataInputStream;
4 import java.io.DataOutputStream;
5 import java.io.IOException;
6 import java.net.InetAddress;
7 import java.net.Socket;
8 import java.util.Scanner;
9
10 public class Client {
11     private InetAddress host;
12     private int port;
13     private int a;
14     private int b;
15     private int c;
16     private int d;
17     private Scanner sc = new Scanner(System.in);
18
19     public Client(InetAddress host, int port) {
20         this.host = host;
21         this.port = port;
22     }
23 }
```

```

24     private void execute() throws IOException {
25         Socket client = new Socket(host, port);
26         DataInputStream dis = new DataInputStream(client.getInputStream());
27         DataOutputStream dos = new DataOutputStream(client.getOutputStream());
28
29         // Nhập dữ liệu
30         a = input("Nhập vào số a: ");
31         b = input("Nhập vào số b: ");
32         c = input("Nhập vào số c: ");
33         d = input("Nhập vào số d: ");
34         dos.writeInt(a);
35         dos.writeInt(b);
36         dos.writeInt(c);
37         dos.writeInt(d);
38
39         while (true) {
40             // Nhận menu và in ra màn hình
41             System.out.println(dis.readUTF());
42
43             int choose = input("Nhập vào tùy chọn của bạn: ");
44             //Gửi tùy chọn đến Server
45             dos.writeInt(choose);
46             switch (choose) {
47                 case 1:
48                     System.out.println("Ước chung lớn nhất = " + dis.readInt());
49                     break;
50                 case 2:
51                     System.out.println("Bội chung nhỏ nhất = " + dis.readInt());
52                     break;
53                 case 3:
54                     System.out.print("Danh sách sau khi sắp xếp tăng dần: ");
55                     System.out.print(dis.readInt() + " ");
56                     System.out.print(dis.readInt() + " ");
57                     System.out.print(dis.readInt() + " ");
58                     System.out.print(dis.readInt() + " ");
59                     break;
60                 case 4:
61                     System.out.print("Danh sách sau khi sắp xếp giảm dần: ");
62                     System.out.print(dis.readInt() + " ");
63                     System.out.print(dis.readInt() + " ");
64                     System.out.print(dis.readInt() + " ");
65                     System.out.print(dis.readInt() + " ");
66                     break;
67                 case 5:
68                     System.out.println("Tổng các số lẻ = " + dis.readInt());
69                     break;
70                 case 6:
71                     System.out.println("Bội chung số chẵn = " + dis.readInt());
72                     break;
73                 case 7:
74                     System.out.println("Các số nguyên tố: ");
75                     int songuyento = dis.readInt();
76                     while(songuyento != -1) {
77                         System.out.print(songuyento + " ");
78                         songuyento = dis.readInt();
79                     }
80                     break;

```

```

82         case 8:
83             System.out.println("Các số chinh phương: ");
84             int sochinhphuong = dis.readInt();
85             while(sochinhphuong != -1) {
86                 System.out.print(sochinhphuong + " ");
87                 sochinhphuong = dis.readInt();
88             }
89             break;
90
91         case 9:
92             System.out.println("Các số hoàn hảo: ");
93             int sohoanhao = dis.readInt();
94             while(sohoanhao != -1) {
95                 System.out.print(sohoanhao + " ");
96                 sohoanhao = dis.readInt();
97             }
98             break;
99         case 10:
100            break;
101
102        default:
103            System.out.println("Vui lòng chỉ nhập số trong menu!");
104            break;
105        }
106        if(choose == 10) break; // Bằng 10 thoát
107    }
108    dis.close();
109    dos.close();
110    client.close();
111}
112
113 public static void main(String[] args) throws IOException {
114     Client client = new Client(InetAddress.getLocalHost(), 15797);
115     client.execute();
116 }
117
118 private int input(String request) {
119     int number = 0;
120     boolean flag = true;
121     do {
122         try {
123             System.out.println(request);
124             number = Integer.parseInt(sc.nextLine());
125             flag = false;
126         } catch (Exception e) {
127             System.out.println("Dữ liệu không đúng định dạng. Vui lòng nhập lại!");
128         }
129     } while (flag);
130     return number;
131 }
132 }
```

* Server

```
3 import java.io.DataInputStream;
4 import java.io.DataOutputStream;
5 import java.io.IOException;
6 import java.net.ServerSocket;
7 import java.net.Socket;
8 import java.sql.SQLException;
9 import java.util.ArrayList;
10 import java.util.Collections;
11
12 public class Server {
13     private int port;
14     private int a;
15     private int b;
16     private int c;
17     private int d;
18
19     public Server(int port) {
20         this.port = port;
21     }
22
23     private void execute() throws IOException{
24         System.out.println("Server is listening...");
25         //Cấu hình Server
26         ServerSocket server = new ServerSocket(port);
27         Socket socket = server.accept();
28         DataInputStream dis = new DataInputStream(socket.getInputStream());
29         DataOutputStream dos = new DataOutputStream(socket.getOutputStream());
30
31         //Nhận dữ liệu từ Client
32         a = dis.readInt();
33         b = dis.readInt();
34         c = dis.readInt();
35         d = dis.readInt();
36         int [] arr = {a,b,c,d};
37
38         while(true) {
39             //Gửi menu về Client
40             dos.writeUTF(menu());
41             //Nhận tùy chọn menu
42             int choose = dis.readInt();
43             switch (choose) {
44                 case 1:
45                     int UCLN = GCD(arr[0], GCD(arr[1], GCD(arr[2], arr[3])));
46                     dos.writeInt(UCLN);
47                     break;
48                 case 2:
49                     dos.writeInt(LCM(arr));
50                     break;
51                 case 3:
52                     ArrayList<Integer> arrASC = ASC(arr); //Mảng sau khi sắp xếp
53                     dos.writeInt(arrASC.get(0));
54                     dos.writeInt(arrASC.get(1));
55                     dos.writeInt(arrASC.get(2));
56                     dos.writeInt(arrASC.get(3));
57                     break;
58             }
59         }
60     }
61 }
```

```

58         case 4:
59             ArrayList<Integer> arrDESC = DESC(arr); //Mảng sau khi sắp xếp
60             dos.writeInt(arrDESC.get(0));
61             dos.writeInt(arrDESC.get(1));
62             dos.writeInt(arrDESC.get(2));
63             dos.writeInt(arrDESC.get(3));
64             break;
65         case 5:
66             dos.writeInt(sumOfOdd(arr));
67             break;
68         case 6:
69             dos.writeInt(sumOfEven(arr));
70             break;
71         case 7:
72             for (Integer item : arr) {
73                 if(soNguyenTo(item) == true) {
74                     dos.writeInt(item);
75                 }
76             }
77             dos.writeInt(-1);      //Đánh dấu kết thúc
78             break;
79
80         case 8:
81             for (Integer item : arr) {
82                 if(soChinhPhuong(item) == true) {
83                     dos.writeInt(item);
84                 }
85             }
86             dos.writeInt(-1);      //Đánh dấu kết thúc
87             break;
88
89         case 9:
90             for (Integer item : arr) {
91                 if(soHoanHao(item) == true) {
92                     dos.writeInt(item);
93                 }
94             }
95             dos.writeInt(-1);      //Đánh dấu kết thúc
96             break;
97         case 10:
98             break;
99
100        default:
101            break;
102        }
103        if(choose == 10) break;
104    }
105    dis.close();
106    dos.close();
107    server.close();
108}
...
109 private String menu() {
110     String menu = "\n\n=====MENU=====\\n"
111             + "1. Tìm ước chung lớn nhất\\n"
112             + "2. Tìm bội chung nhỏ nhất\\n"
113             + "3. Sắp xếp tăng dần\\n"
114             + "4. Sắp xếp giảm dần\\n"
115             + "5. Tổng tất cả các số lẻ\\n"
116             + "6. Tổng tất cả các số chẵn\\n"
117             + "7. Các số nguyên tố\\n"
118             + "8. Các số chính phương\\n"
119             + "9. Các số hoàn hảo\\n"
120             + "10. Thoát";
121     return menu;
122 }
123

```

```

124     private int GCD(int a, int b) {
125         while (a != b) {
126             if (a > b)
127                 a = a - b;
128             else
129                 b = b - a;
130         }
131         return a;
132     }
133
134     private int LCM(int [] arr)
135     {
136         int max = 0, k = 1;
137         for (int i = 0; i < arr.length; i++) { //Tim số lớn nhất trong mảng
138             if (max < arr[i]) {
139                 max = arr[i];
140             }
141         }
142         int temp = max;
143         for (int i = 0; i < arr.length; i++) {
144             while (max % arr[i] != 0) {
145                 k++;
146                 max = temp * k;
147             }
148         }
149         return max;
150     }
151
152     private ArrayList<Integer> ASC (int [] arr) {
153         ArrayList<Integer> list = new ArrayList<>();
154         for (int i : arr) {
155             list.add(i);
156         }
157         Collections.sort(list);
158         return list;
159     }
160
161     private ArrayList<Integer> DESC (int [] arr) {
162         ArrayList<Integer> list = new ArrayList<>();
163         for (int i : arr) {
164             list.add(i);
165         }
166         Collections.sort(list);
167         Collections.reverse(list);
168         return list;
169     }
170
171     private int sumOfOdd(int [] arr) {
172         int sum = 0;
173         for (int i : arr) {
174             if(i % 2 == 1) {
175                 sum += i;
176             }
177         }
178         return sum;
179     }
180

```

```

181     private int sumOfEven(int [] arr) {
182         int sum = 0;
183         for (int i : arr) {
184             if(i % 2 == 0) {
185                 sum += i;
186             }
187         }
188         return sum;
189     }
190
191     private boolean soNguyenTo(int n) {
192         if(n < 2) return false;
193         for (int i = 2; i <= Math.sqrt(n); i++) {
194             if((n%i == 0) && (n != i)) return false;
195         }
196         return true;
197     }
198
199     private boolean soChinhPhuong(int n) {
200         int temp = (int)Math.sqrt(n);
201         return (temp*temp == n);
202     }
203
204     private boolean soHoanHao(int n) {
205         int tong = 0;
206         for (int i = 1; i < n ; i++){
207             if ( n % i == 0 ) tong = tong + i;
208         }
209         return (tong == n);
210     }
211
212     public static void main(String[] args) throws IOException {
213         Server server = new Server(15797);
214         server.execute();
215     }
216
217 }
```

7. Từ điển

* *Client*

```

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.net.Socket;

public class TCPClient {

    private static Socket socket;
    private static BufferedReader in;
    private static BufferedWriter out;
    private static BufferedReader stdIn;
    private static boolean loop = true;

    public TCPClient() {
    }

    //Hàm xử lý
    public static void handle() {
        try {
            socket = new Socket("localhost", 2001);
            System.out.println("Khach hang da ket noi...");

            in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
            out = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()));

            stdIn = new BufferedReader(new InputStreamReader(System.in));

            while (loop) {
                System.out.println("Vui long nhap tu tra cuu: ");
                String input = stdIn.readLine();
                out.write(input);
                out.newLine();
                out.flush();

                String message = in.readLine();
                if (message.equals("Ngat ket noi....")) {
                    loop = false;
                }
                System.out.println(message);
            }
            in.close();
            stdIn.close();
            out.close();
            socket.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public static void main(String[] args) {
        TCPClient.handle();
    }
}

```

*Server

```
import java.io.*;
import java.net.ServerSocket;
import java.net.Socket;
import java.nio.ByteBuffer;
import java.nio.charset.StandardCharsets;
import java.nio.file.Paths;
import java.util.*;

public class TCPServer {

    private static ServerSocket serverSocket;
    private static Socket socket;
    private static BufferedReader in;
    private static BufferedWriter out;
    private static LinkedHashMap<String, String> dictionaries = new LinkedHashMap<>();
    private static boolean loop = true;
    private static final String URL_FILE = "./src/bail/dictionary.txt";

    private static void readFile(String filePath) {
        try {
            Scanner readFile = new Scanner(new File(filePath));
            while (readFile.hasNextLine()) {
                String[] line = readFile.nextLine().split(";");
                String key = line[0];
                String value = line[1];
                dictionaries.putIfAbsent(key, value);
            }
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        }
    }

    public static void main(String[] args) {
        TCPServer.readFile(URL_FILE);
        try {
            serverSocket = new ServerSocket(2001);
            System.out.println("May chu dang chay tren cong 2001...");
            socket = serverSocket.accept();
            System.out.println("Khach hang da ket noi!");

            in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
            out = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()));

            while (loop) {
                String[] line = in.readLine().split(";");
                String note = line[0];
                System.out.println("Khach hang muon tra cuu: " + note);

                switch (note.toUpperCase()) {
                    case "BYE":
                        loop = false;
                        out.write("Ngat ket noi....");
                        out.newLine();
                        out.flush();
                        break;
                    case "ADD":
                        String x = line[1];
                        String y = line[2];
                        boolean flagAdd = false;
                        for (String key : dictionaries.keySet()) {
                            if (key.toLowerCase().equals(x.toLowerCase())) {
                                out.write("Da ton tai trong tu dien");
                                out.newLine();
                                out.flush();
                                flagAdd = true;
                                break;
                            }
                        }
                }
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

```

        }
        if (!flagAdd) {
            dictionaries.putIfAbsent(x, y);
            PrintWriter printWriter = new PrintWriter(new FileWriter(URL_FILE), true);
            for (String key : dictionaries.keySet()) {
                printWriter.write(key + ";" + dictionaries.get(key) + "\n");
            }
            printWriter.flush();
            readFile(URL_FILE);
            out.write("Them vao tu dien thanh cong");
            out.newLine();
            out.flush();
        }
        break;
    case "DEL":
        if (dictionaries.containsKey(line[1])) {
            dictionaries.remove(line[1]);
            PrintWriter printWriter = new PrintWriter(new FileWriter(URL_FILE), true);
            for (String key : dictionaries.keySet()) {
                printWriter.write(key + ";" + dictionaries.get(key) + "\n");
            }
            printWriter.flush();
            readFile(URL_FILE);
            out.write("Xoa thanh cong");
            out.newLine();
            out.flush();
        } else {
            out.write("Khong ton tai tu do trong tu dien");
            out.newLine();
            out.flush();
        }
        break;
    default:
        //Tieng anh sang tieng viet
        boolean flag = false;
        for (String key : dictionaries.keySet()) {
            if (note.equalsIgnoreCase(key)) {
                String value = dictionaries.get(key);
                out.write("dich ra tieng anh:" + value);
                out.newLine();
                out.flush();
                flag = true;
                break;
            }
        }
        //Tieng viet sang tieng anh
        if (!flag) {
            for (String key : dictionaries.keySet()) {
                String value = dictionaries.get(key);
                if (note.equalsIgnoreCase(value)) {
                    out.write("dich ra tieng anh:" + key);
                    out.newLine();
                    out.flush();
                    flag = true;
                    break;
                }
            }
        }
        if (!flag) {
            out.write("Khong tim thay tu can tra cuu! Vui long nhap lai.");
            out.newLine();
            out.flush();
            break;
        }
    }
}
}

in.close();
out.close();
socket.close();
serverSocket.close();
} catch (IOException e) {
    System.out.println(e.getMessage());
}
}
}

```

8. Chuỗi đảo ngược TCP

Viết chương trình gửi tin nhắn hai chiều giữa client-server sử dụng TCP socket.

Client gửi 1 chuỗi ký tự bất kỳ đến server

Server nhận và gửi chuỗi đảo ngược về client

Client xuất kết quả ra console, chương trình kết thúc khi client gửi chuỗi bye.

* **Client:**

```

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.net.Socket;

public class client {
    Socket socket = null;
    BufferedWriter out = null;
    BufferedReader in = null;
    BufferedReader stdIn = null;

    public client(String host, int port){
        try{
            socket = new Socket(host, port);
            System.out.println("Client connected");
            out = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()));
            in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
            stdIn = new BufferedReader(new InputStreamReader(System.in));
            String line="";
            while(!line.equals("bye")){
                try{
                    //nhập chuỗi
                    line = stdIn.readLine();
                    System.out.println("Client sent:" + line);
                    out.write(line);
                    out.newLine();
                    out.flush(); // đẩy dữ liệu qua server

                    //nhận kết quả trả về từ server
                    String da= in.readLine();
                    System.out.println("Result:" +da);
                }catch(Exception e){
                    System.err.println(e);
                }
            }
            in.close();
            stdIn.close();
            out.close();
            socket.close();
        }catch(Exception e){
            System.out.println(e);
        }
    }

    public static void main(String[] args){
        client client = new client("localhost",6000);
    }
}

```

*Server

```
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.net.ServerSocket;
import java.net.Socket;

public class server {
    ServerSocket server = null;
    Socket socket =null;
    BufferedWriter out =null;
    BufferedReader in =null;
    StringBuffer stringBuffer= null;
    public server(int port){
        try{
            server =new ServerSocket(port);
            System.out.println("Server started. Waitting for client....");
            socket =server.accept();
            System.out.println("Client" + socket.getInetAddress() + "is connect");
            out = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()));
            in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
            String line ="";
            while(!line.equals("bye")){
                try{
                    //nhận chuỗi từ Client
                    line = in.readLine();
                    System.out.println("Server received: " + line);
                    //xử lý đảo chuỗi
                    stringBuffer = new StringBuffer(line);
                    String result = stringBuffer.reverse().toString();
                    out.write(result);
                    out.newLine();
                    out.flush(); // đẩy dữ liệu qua
                }catch(Exception e){
                    System.out.println(e);
                }
            }
            in.close();
            out.close();
            socket.close();
            server.close();
        } catch (Exception e){
            System.err.println(e);
        }
    }
    public static void main(String[] args){
        server server = new server(6000);
    }
}
```

9. Số hoàn hảo TCP

Viết chương trình tìm số hoàn hảo, hoạt động theo mô hình client-server, sử dụng TCP socket
Client gửi 1 số n nguyên dương đến server.

Server kiểm tra n, nếu:

- o Là số hoàn hảo: trả kết quả về client và xuất ra màn hình
- o Không phải số hoàn hảo: trả về client số hoàn hảo lớn hơn và gần n nhất.

* Client

```

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.net.Socket;

public class client {
    Socket socket = null;
    BufferedWriter out = null;
    BufferedReader in = null;
    BufferedReader stdIn = null;

    public client(String host, int port){
        try{
            socket = new Socket(host, port);
            System.out.println("Client connected");
            out = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()));
            in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
            stdIn = new BufferedReader(new InputStreamReader(System.in));
            String line="";
            while(!line.equals("bye")){
                try{
                    //nhập số
                    line = stdIn.readLine();
                    System.out.println("Client sent:" + line);
                    out.write(line);
                    out.newLine();
                    out.flush(); // đẩy dữ liệu qua server

                    //nhận kết quả đào chuỗi từ server
                    String da= in.readLine();
                    System.out.println("Result:" +da);
                }catch(Exception e){
                    System.err.println(e);
                }
            }
            in.close();
            stdIn.close();
            out.close();
            socket.close();
        }catch(Exception e){
            System.out.println(e);
        }
    }

    public static void main(String[] args){
        client client = new client("localhost",6000);
    }
}

```

*Server

```
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.net.ServerSocket;
import java.net.Socket;

public class server {
    ServerSocket server = null;
    Socket socket =null;
    BufferedWriter out =null;
    BufferedReader in =null;
    StringBuffer stringBuffer= null;
    public int ktSoHoanHao(int n){
        int s=0;
        for(int i=1;i<n;i++){
            if(n%i==0)
                s+=i;
        }
        if(s==n)
            return 1;
        return 0;
    }
    public int soHHKeSau(int n){
        for(int i=n+1;i<Integer.MAX_VALUE;i++){
            if(ktSoHoanHao(i)==1)
                return i;
        }
        return 0;
    }
    public server(int port){
        try{
            server =new ServerSocket(port);
            System.out.println("Server started. Waitting for client....");
            socket =server.accept();
            System.out.println("Client" + socket.getInetAddress()+" is connect");
            out = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()));
            in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
            String line ="";
            while(!line.equals("bye")){
                try{
                    //nhận số từ Client
                    line = in.readLine();
                    int a = Integer.parseInt(line);
                    System.out.println("Server received: " + a);
                    if(ktSoHoanHao(a)==1){
                        out.write(a +" là số hoàn hảo");
                    }
                    else{
                        out.write(a+" không phải là số hoàn hảo"+a+" nhat la: "+soHHKeSau(a));
                    }
                    out.newLine();
                    out.flush();// đẩy dữ liệu qua
                }catch(Exception e){
                    System.out.println(e);
                }
            }
            in.close();
            out.close();
            socket.close();
            server.close();
        } catch (Exception e){
            System.err.println(e);
        }
    }
    public static void main(String[] args){
        server server = new server(6000);
    }
}
```

10. Phân tích thừa số nguyên tố:

Viết chương trình phân tích số, hoạt động theo mô hình client-server, sử dụng TCP socket

- Client gửi số nguyên dương $n \geq 10$ đến server.
- Server phân tích n thành tích các số nguyên tố và gửi trả ngược lại client

- Client xuất kết quả ra console

**Client:*

```
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.net.Socket;

public class client {
    Socket socket = null;
    BufferedWriter out = null;
    BufferedReader in = null;
    BufferedReader stdIn = null;

    public client(String host, int port){
        try{
            socket = new Socket(host, port);
            System.out.println("Client connected");
            out = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()));
            in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
            stdIn = new BufferedReader(new InputStreamReader(System.in));
            String line="";
            while(!line.equals("bye")){
                try{
                    //nhập số căn phân tích
                    line = stdIn.readLine();
                    int a = Integer.parseInt(line);
                    if(a<=10){
                        System.out.println("Nhập lại:");
                        line = stdIn.readLine();
                    }
                    System.out.println("Client sent:" + line);
                    out.write(line);
                    out.newLine();
                    out.flush(); // đẩy dữ liệu qua server

                    //nhận kết quả phân tích từ server
                    String da= in.readLine();
                    System.out.println("Result:" +da);
                }catch(Exception e){
                    System.err.println(e);
                }
            }
            in.close();
            out.close();
            stdIn.close();
            socket.close();
        }catch(Exception e){
            System.out.println(e);
        }
    }

    public static void main(String[] args){
        client client = new client("localhost",6000);
    }
}
```

**Server*

```

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import static java.lang.Math.sqrt;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.ArrayList;
import java.util.List;

public class server {
    ServerSocket server = null;
    Socket socket =null;
    BufferedWriter out =null;
    BufferedReader in =null;
    StringBuffer stringBuffer= null;
    /*public static List<Integer> phantichSNT(int n){
        int i=2;
        List<Integer> list = new ArrayList<Integer>();
        //phân tích
        while(n>1){
            if(n%i==0){
                n=n/i;
                list.add(i);
            }
            else{
                i++;
            }
        }
        //nếu list trống thì add n vào
        if(list.isEmpty()){
            list.add(n);
        }
        return list;
    }*/
    public static boolean ktrsnt(int n){
        if(n<=1)
            return false;
        for(int i=2; i<=sqrt(n);i++){
            if(n%i==0)
                return false;
        }
        return true;
    }
    public server(int port){
        try{
            server =new ServerSocket(port);
            System.out.println("Server started. Waiting for client....");
            socket =server.accept();
            System.out.println("Client" + socket.getInetAddress()+" is connect");
            out = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()));
            in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
            String line ="";
            while(!line.equals("bye")){
                //nhận số từ Client
                line = in.readLine();
                int a = Integer.parseInt(line);
                System.out.println("Server received: " + a);
                //xử lý phân tích thành các số nguyên tố
                /*List<Integer> list= phantichSNT(a);
                int size= list.size();
                for(int i=0;i<size-1;i++){
                    out.write(list.get(i)+ "x");
                }*/
                for(int i=2;i<=a;i++){
                    while(a%i==0 && ktrsnt(i)){
                        out.write(i+"x");
                        a=a/i;
                    }
                }
            }
        }
    }
    public static void main(String[] args){
        server server = new server(6000);
        try{
            server.start();
        } catch (Exception e){
            System.err.println(e);
        }
    }
}

```

11. Viết chương trình đoán số, hoạt động theo mô hình client-server, sử dụng TCP socket

- Khi client kết nối, server tạo sẵn 1 số nguyên ngẫu nhiên $n \leq 100$
- Client đoán số do server tạo, nếu không đúng, server cần gợi ý bằng cách cho biết số client gửi lớn hơn hay nhỏ hơn n .
- Quá trình lặp liên tục cho tới khi client gửi đúng số $= n$. Server xuất các thông kê: số lần client đoán, tổng thời gian đoán.

*Client

```
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.net.Socket;

public class BT4Client {
    Socket socket = null;
    BufferedWriter out = null;
    BufferedReader in = null;
    BufferedReader stdIn = null;

    public BT4Client(String host, int port){
        try{
            socket = new Socket(host, port);
            System.out.println("Client connected");
            System.out.println("Hay doan 1 con so");
            out = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()));
            in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
            stdIn = new BufferedReader(new InputStreamReader(System.in));
            String line ="";
            String.newLine = " ";
            //String random=in.readLine();
            while(!line.equals(newLine)){
                line = stdIn.readLine();
                try{
                    int a=Integer.parseInt(line);
                    if(a<=100){
                        System.out.println("Client sent: "+line);
                        out.write(line);
                        out.newLine();
                        out.flush();

                        newLine=in.readLine();
                        System.out.println("Client received: "+newLine);
                    }
                    else{
                        System.out.println("Vui long nhap so <=100.");
                    }
                }catch(NumberFormatException e){
                    System.out.println("Vui long nhap so <=100.");
                }
            }
            newLine=in.readLine();
            System.out.println("Client received: "+newLine);
            in.close();
            out.close();
            stdIn.close();
            socket.close();
        }catch(Exception e){
            System.err.println(e);
        }
    }

    public static void main (String[] args){
        BT4Client client = new BT4Client("localhost", 6000);
    }
}
```

* Server

```

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import static java.lang.Math.sqrt;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.HashMap;
import java.util.Map;
import java.util.Random;
import java.util.Set;
import java.util.StringTokenizer;

public class BT4Server {
    ServerSocket server = null;
    Socket socket = null;
    BufferedWriter out = null;
    BufferedReader in = null;

    public BT4Server(int port){
        try{
            server = new ServerSocket(port);
            System.out.println("Server started. Waiting for client.....");
            socket = server.accept();
            System.out.println("Client " +socket.getInetAddress()+" is connected.");
            out = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()));
            in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
            Random generator = new Random();
            int n=generator.nextInt(100)+1;
            System.out.println("So ngau nhien: "+n);
            String line="";
            String.newLine="";
            while(!line.equals(String.valueOf(n))){
                line = in.readLine();
                int t=Integer.parseInt(line);
                System.out.println("Server received: "+line);
                if(n<t)
                    newLine="Sai. Goi y: So can doan be hon so nay.";
                else
                    newLine="Sai. Goi y: So can doan lon hon so nay.";
                System.out.println("Server sent: "+newLine);
                out.write(newLine);
                out.newLine();
                out.flush();
            }
            newLine="Chinh xac!";
            System.out.println("Server sent: "+newLine);
            /*out.write(newLine);
            out.newLine();
            out.flush();*/
            in.close();
            out.close();
            socket.close();
            server.close();
        }catch(Exception e){
            System.err.println(e);
        }
    }

    public static void main (String[] args){
        BT4Server server = new BT4Server (6000);
    }
}

```

12. Viết chương trình tính toán, hoạt động theo mô hình client-server, sử dụng TCP socket

- Client gửi 1 chuỗi phép toán gồm 2 số và 1 trong 4 phép toán (+, -, *, /) đến server
- Server phân tích chuỗi, tính kết quả và trả lại client hoặc trả thông báo lỗi nếu chuỗi phép toán không đúng format.
- Client xuất kết quả ra console

* **Client**

```

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.net.Socket;

public class BT5Client {
    Socket socket = null;
    BufferedWriter out = null;
    BufferedReader in = null;
    BufferedReader stdIn = null;

    public BT5Client(String host, int port){
        try{
            socket = new Socket(host, port);
            System.out.println("Client connected");
            out = new OutputStreamWriter(socket.getOutputStream());
            in = new InputStreamReader(socket.getInputStream());
            stdIn = new BufferedReader(new InputStreamReader(System.in));
            String line ="";
            String.newLine="";
            while(!line.equals("bye")){
                line = stdIn.readLine();

                System.out.println("Client sent: "+line);
                out.write(line);
                out.newLine();
                out.flush();

                newLine=in.readLine();
                System.out.println("Client received: "+newLine);

            }
            in.close();
            out.close();
            stdIn.close();
            socket.close();
        }catch(Exception e){
            System.err.println(e);
        }
    }

    public static void main (String[] args){
        BT5Client client = new BT5Client("localhost", 6000);
    }
}

```

* Server

```
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import static java.lang.Math.sqrt;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.HashMap;
import java.util.Map;
import java.util.Set;
import java.util.StringTokenizer;

public class BT5Server {
    ServerSocket server = null;
    Socket socket = null;
    BufferedWriter out = null;
    BufferedReader in = null;

    public BT5Server(int port){
        try{
            server = new ServerSocket(port);
            System.out.println("Server started. Waiting for client.....");
            socket = server.accept();
            System.out.println("Client " +socket.getInetAddress()+" is connected.");
            out = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()));
            in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
            String line="";
            while(!line.equals("bye")){
                line = in.readLine();
                String newLine="";
                System.out.println("Server received: "+line);
                String pattern="\\"d{0,6}[*/-]\\d{0,6}";
                if(line.matches(pattern)==false){
                    newLine="Khong dung format";
                }
                else{
                    int a,b;
                    double result = 0;
                    String op;
                    StringTokenizer st = new StringTokenizer(line, "+-*/");
                    a=Integer.parseInt(st.nextToken());
                    op=st.nextToken();
                    b=Integer.parseInt(st.nextToken());

                    switch(op){
                        case "+":{
                            newLine+=a+b;
                            break;
                        }
                        case "-":{
                            newLine+=a-b;
                            break;
                        }
                        case "*":{
                            newLine+=a*b;
                            break;
                        }
                        case "/":{
                            newLine+=a*1.0/b;
                            break;
                        }
                    }
                }
                System.out.println("Server sent: "+newLine);
                out.write(newLine);
                out.newLine();
                out.flush();
            }
            System.out.println("Closing connection");
            in.close();
            out.close();
            socket.close();
            server.close();
        }catch(Exception e){
            System.err.println(e);
        }
        public int kt_snt(long n){
            if(n<2)
                return 0;
            double k=sqrt(n);
            for(int i=2;i<=k;i++)
                if(n%i==0)
                    return 0;
            return 1;
        }
    }
    public static void main (String[] args){
        BT5Server server = new BT5Server (6000);
    }
}
```

13. Viết chương trình tìm thông tin IP, hoạt động theo mô hình client-server, sử dụng TCP socket.

- Tra thông tin server (chỉ dùng khi server hoạt động ở một mạng có NAT): client gửi lệnh hello đến server. Server trả lại client public IP và private IP của server.
- Tra cứu IP: client gửi lệnh req x, với x là một địa chỉ IP public. Server trả lại client các thông tin về IP x gồm: thành phố - quốc gia – châu lục mà IP đó thuộc về hoặc trả về thông báo lỗi nếu IP không đúng format/IP private.
- Trường hợp client gửi không đúng cú pháp, server trả về hướng dẫn sử dụng chương trình.

* Client

```
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.net.Socket;

public class clientbai2 {
    Socket socket = null;
    BufferedWriter out = null;
    BufferedReader in = null;
    BufferedReader stdIn = null;

    public clientbai2(String host, int port){
        try{
            socket = new Socket(host, port);
            System.out.println("Client connected");
            out = new OutputStreamWriter(socket.getOutputStream());
            in = new InputStreamReader(socket.getInputStream());
            stdIn = new BufferedReader(new InputStreamReader(System.in));
            String line="";
            while(!line.equals("bye")){
                try{
                    //nhập dữ liệu
                    line = stdIn.readLine();

                    System.out.println("Client sent:" + line);
                    out.write(line);
                    out.newLine();
                    out.flush();// đẩy dữ liệu qua server

                    //nhận kết quả từ server
                    String da= in.readLine();
                    System.out.println(da);
                    System.out.println("-----");
                }catch(Exception e){
                    System.err.println(e);
                }
            }
            in.close();
            out.close();
            stdIn.close();
            socket.close();
        }catch(Exception e){
            System.out.println(e);
        }
    }

    public static void main(String[] args){
        clientbai2 client = new clientbai2("localhost",6000);
    }
}
```

* Server

```

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.net.ServerSocket;
import java.net.Socket;
import java.net.URL;

public class serverbai2 {
    ServerSocket server = null;
    Socket socket =null;
    BufferedWriter out =null;
    BufferedReader in =null;
    StringBuffer stringBuffer= null;
    BufferedReader in_server_pub_IP = null;
    BufferedReader in_info_IP = null;
    public serverbai2(int port){
        try{
            server = new ServerSocket(port);
            while(true){
                try{
                    System.out.println("Server started. Waiting for client.....");
                    socket = server.accept();
                    System.out.println("Client " +socket.getInetAddress().getHostAddress()+" is connected.");
                    out = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()));
                    in = new BufferedReader(new InputStreamReader(socket.getInputStream()));

                    while(true){
                        String line = in.readLine();
                        String.newLine="";
                        if(line.equals("bye"))
                            break;
                        System.out.println("Server received: "+line);
                        //client gui "hello"
                        if(line.equals("hello")){
                            URL pubIP = new URL("http://checkip.amazonaws.com");
                            in_server_pub_IP = new BufferedReader(new InputStreamReader(pubIP.openStream()));
                            String ip = in_server_pub_IP.readLine();
                            newLine="Server's public IP: "+ip+" - Server's private IP: "+server.getInetAddress().getHostAddress() ;
                            System.out.println(newLine);
                        }
                        else{
                            String w1 = line.substring(0,3); //lay chu req
                            if(!w1.equals("req")||line.charAt(3)!=' ')
                                newLine="Không đúng cú pháp....";
                            else{
                                //client gui req x
                                try{
                                    String w2 = line.substring(4,7);
                                    String w3 = line.substring(4,11);
                                    String w4 = line.substring(4); //lay dia chi ip
                                    String pattern="\d{1,3}.\d{1,3}.\d{1,3}.\d{1,3}";
                                    if(w4.matches(pattern)==false){
                                        newLine="IP này không đúng format";
                                    }
                                    newLine= IP hay không đúng format ;
                                }
                                else{
                                    if(w2.equals("10.")||w3.equals("192.168")||w3.equals("172.16."))
                                        newLine="Đây là private IP.";
                                    else{
                                        URL infoIP = new URL("http://ip-api.com/line/"+w4+"?fields=status,message,country,city,timezone,query");
                                        in_info_IP = new BufferedReader(new InputStreamReader(infoIP.openStream()));
                                        String status = in_info_IP.readLine();
                                        if(status.equals("success")){
                                            String country = in_info_IP.readLine();
                                            String city = in_info_IP.readLine();
                                            String timezone = in_info_IP.readLine();
                                            String continent = "";
                                            for(int i=0;i<timezone.length();i++){
                                                if(timezone.charAt(i)=='/'){
                                                    continent=timezone.substring(0,i);
                                                    break;
                                                }
                                            }
                                            newLine="Country: "+country+" - City: "+city+" - Continent: "+continent;
                                        }
                                        else System.out.println("IP không tồn tại");
                                    }
                                }
                            }
                        }
                    }catch(Exception e){
                        newLine="Không đúng cú pháp.";
                    }
                }
            }
        }
    }
}

```

```

        ,
        System.out.println("Server sent: "+newLine);
        out.write(newLine);
        out.newLine();
        out.flush();
    }
    System.out.println("Server closed connection");
    in.close();
    in_server_pub_IP.close();
    out.close();
    socket.close();
    //server.close();
}catch(IOException e){
    System.out.println("Unexpected connection close "+socket.getInetAddress().getHostAddress());
}
}
}catch(Exception e){
    System.err.println(e);
}
}
public static void main(String[] args){
    serverbai2 server = new serverbai2(6000);
}
}
}

```

14. Viết chương trình tra cứu thông tin cá nhân, hoạt động theo mô hình client-server, sử dụng TCP socket

- Client gửi 1 số chứng minh nhân dân/căn cước công dân người VN đến server.
- Server tìm kiếm họ tên, quê quán tương ứng với số CMND/CCCD đó và gửi trả ngược lại client hoặc trả thông báo lỗi nếu không tìm thấy thông tin.
- Client xuất kết quả ra console.

* Client

```

import java.net.*;
import java.io.*;
public class Clientbai4 {
    Socket socket = null;
    BufferedReader in = null;
    BufferedWriter out = null;
    BufferedReader stdIn = null;

    public Clientbai4(String host, int port){
        try{
            socket = new Socket(host,port);
            System.out.println("Client connected.");
            out = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()));
            in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
            stdIn = new BufferedReader(new InputStreamReader(System.in));
            String line = "";
            while(!line.equals("bye")){
                try{
                    System.out.println("Nhập CMND: ");
                    line = stdIn.readLine();
                    System.out.println("CMND: "+line);
                    out.write(line);
                    out.newLine();
                    out.flush(); //đẩy dữ liệu qua Server
                }catch(Exception e){
                    System.out.println("Nhận kết quả từ Server");
                    String data = in.readLine();
                    System.out.println("Result: "+data);
                }
            }
            in.close();
            out.close();
            stdIn.close();
            socket.close();
        }catch(Exception e){
            System.out.println(e);
        }
    }

    public static void main(String[] args){
        Clientbai4 client = new Clientbai4("localhost",6000);
    }
}

```

* Server

```
import java.io.*;
import java.net.*;
import org.jsoup.Jsoup;
import org.jsoup.nodes.Document;
import org.jsoup.select.Elements;

public class Serverbai4 {
    ServerSocket server = null;
    Socket socket = null;
    BufferedReader in = null;
    BufferedWriter out = null;
    StringBuffer stringBuffer;

    public Serverbai4(int port){
        try{
            server = new ServerSocket(port);
            System.out.println("Server starting.....");
            socket = server.accept();
            System.out.println("Client accepted");
            out = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()));
            in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
            String line="";
            while(!line.equals("bye")){
                try{
                    //nhận chuỗi từ Client
                    line = in.readLine();
                    System.out.println("Server received: "+line);
                    String linesent = "";
                    if (line.equals("bye")) {
                        linesent = "stop";
                    } else {
                        String[] result = GetCMND(Long.valueOf(line));
                        System.out.println("");
                        String sb = "";
                        for (int j = 0; j < result.length; j++) {
                            sb+= result[j];
                            sb+="; ";
                        }
                        linesent = sb;
                    }
                    System.out.println("Sever send: " + linesent);
                    out.write(linesent);
                    out.newLine();
                    out.flush(); //đẩy dữ liệu qua Client
                }catch(Exception e){
                    System.err.println(e);
                }
                in.close();
                out.close();
                socket.close();
                server.close();
            }catch(Exception e){
                System.err.println(e);
            }
        }
        public String[] GetCMND(long cmnd){
            String[] res = null;
            try {
                // neu tim CMND khong co = loi
                String url = Jsoup.connect("https://masothue.com/Search/?q=" + cmnd + "&type=auto&force-search=1")
                    .followRedirects(true) // cho phep di
                    .execute()
                    .url()
                    .toExternalForm(); // convert sang String
                Document doc = Jsoup.connect(url).get();

                Elements info = doc.getElementsByClass("copy");
                res = new String[info.size()];
                for(int i = 0 ; i < info.size(); i++){
                    res[i] = info.get(i).text();
                    System.out.println(""+res[i]);
                }
            } catch (IOException e) {
                System.err.println(e);
            }
            return res;
        }
        public static void main(String[] args){
            Serverbai4 server = new Serverbai4(6000);
        }
    }
}
```

15. Viết chương trình tính toán, hoạt động theo mô hình client-server, sử dụng TCP socket

□ Client gửi 1 chuỗi phép toán gồm nhiều số phân cách nhau bởi 1 trong 4 phép toán (+, -, *, /) đến server, giả sử chuỗi phép toán không chứa các dấu ngoặc. Ví dụ chuỗi phép toán sau: 12+34-56*78/4+14-17

□ Server phân tích chuỗi, tính kết quả và trả lại client hoặc trả thông báo lỗi nếu chuỗi phép toán không đúng format.

□ Client xuất kết quả ra console

* *Client*

```
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.net.Socket;

public class clientbai3 {
    Socket socket = null;
    BufferedWriter out = null;
    BufferedReader in = null;
    BufferedReader stdIn = null;

    public clientbai3(String host, int port){
        try{
            socket = new Socket(host, port);
            System.out.println("Client connected");
            out = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()));
            in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
            stdIn = new BufferedReader(new InputStreamReader(System.in));
            String line ="";
            String.newLine ="";
            while(!line.equals("bye")){
                line = stdIn.readLine();

                System.out.println("Client sent: "+line);
                out.write(line);
                out.newLine();
                out.flush();

                newLine=in.readLine();
                System.out.println(newLine);

            }
            in.close();
            out.close();
            stdIn.close();
            socket.close();
        }catch(Exception e){
            System.err.println(e);
        }
    }

    public static void main (String[] args){
        clientbai3 client = new clientbai3("localhost", 6000);
    }
}
```

* *Server*

```

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import static java.lang.Math.sqrt;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;
import java.util.Scanner;
import java.util.Set;
import java.util.StringTokenizer;

public class severbai3 {
    ServerSocket server = null;
    Socket socket = null;
    BufferedWriter out =null;
    BufferedReader in = null;
    private int result;
    private ArrayList<String> ListNumber;
    private ArrayList<String> ListCal;
    private String stCalculation;

    public severbai3(String stCalculation){
        this.stCalculation = stCalculation;
    }
    public int Tinh(String stCalculation){
        ArrayList ListNumber=new ArrayList<>();
        ArrayList ListCal=new ArrayList<>();
        StringTokenizer st = new StringTokenizer(stCalculation, "+-*/",true);
        while(st.hasMoreTokens()){
            ListNumber.add(st.nextToken());
            if(st.hasMoreTokens()){
                ListCal.add(st.nextToken());
            }
        }
        //Tinh phep nhan chia truoc
        for(int i=0; i<ListCal.size();i++){
            if(ListCal.get(i).toString().equals("*")){
                int t = Integer.parseInt((String) ListNumber.get(i))*Integer.parseInt((String) ListNumber.get(i+1));
                ListCal.remove(i);
                ListNumber.set(i, t+"");
                ListNumber.remove(i+1);
                i=0;
            }
            if(ListCal.get(i).toString().equals("/")){
                int t1 = Integer.parseInt((String) ListNumber.get(i))/Integer.parseInt((String) ListNumber.get(i+1));
                ListCal.remove(i);
                ListNumber.set(i, t1+"");
                ListNumber.remove(i+1);
                i=0;
            }
        }
        . . .
        //Tinh phep cong tru
        for(int i=0; i<ListCal.size();i++){
            if(ListCal.get(i).toString().equals("+")){
                int t = Integer.parseInt((String) ListNumber.get(i))+Integer.parseInt((String) ListNumber.get(i+1));
                ListCal.remove(i);
                ListNumber.set(i, t+"");
                ListNumber.remove(i+1);
                i=0;
            }
            if(ListCal.get(i).toString().equals("-")){
                int t1 = Integer.parseInt((String) ListNumber.get(i))-Integer.parseInt((String) ListNumber.get(i+1));
                ListCal.remove(i);
                ListNumber.set(i, t1+"");
                ListNumber.remove(i+1);
                i=0;
            }
        }
        return result = Integer.parseInt((String) ListNumber.get(0));
    }
}

```

```

public severbai3(int port){
    try{
        server = new ServerSocket(port);
        System.out.println("Server started. Waiting for client.....");
        socket = server.accept();
        System.out.println("Client " +socket.getInetAddress()+" is connected.");
        out = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()));
        in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
        String line="";
        while(!line.equals("bye")){
            line = in.readLine();
            String newLine="";
            System.out.println("Server received: "+line);
            int kq=Tinh(line);
            //String pattern="\d{0,6}[*+-]\d{0,6}";
            //if(line.matches(pattern)==false){
            //    newLine="Khong dung format";
            //}
            //else{
            /*int a,b;
            double result = 0;
            String op*/
            // StringTokenizer st = new StringTokenizer(line, "-*/",true);
            /*a=Integer.parseInt(st.nextToken());
            op=st.nextToken();
            b=Integer.parseInt(st.nextToken());

            switch(op){
                case "+":{
                    newLine+=a+b;
                    break;
                }
                case "-":{
                    newLine+=a-b;
                    break;
                }
                case "*":{
                    newLine+=a*b;
                    break;
                }
                case "/":{
                    newLine+=a*1.0/b;
                    break;
                }
            }*/
            */
            //System.out.println("Server sent: "+newLine);
            out.write("Kết quả:" +kq);
            out.newLine();
            out.flush();
        }
        /*public int kt_snt(long n){
            if(n<2)
                return 0;
            double k=sqrt(n);
            for(int i=2;i<=k;i++)
                if(n%i==0)
                    return 0;
            return 1;
        }*/
        public static void main (String[] args){
            severbai3 server = new severbai3 (6000);
        }
    }
}

```

16. Viết chương trình tra thông tin và điểm sinh viên trên cổng thông tin đào tạo, sử dụng mô hình client-server, dựa trên TCPSocket.

- Client gửi 1 mã số sinh viên đến server và chờ nhận kết quả phản hồi.
- Server thực hiện 2 chức năng dưới đây và gửi dữ liệu về lại client:
 - o Chức năng 1: tra cứu họ tên, nơi sinh, ngành, khóa đào tạo.

o Chức năng 2: điểm chi tiết học kỳ 1 năm học 2019-2020 (gồm: mã học phần, tên học phần, điểm kết thúc học phần – hệ 10)

*Client

```
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.net.Socket;

public class clientbai1 {
    Socket socket = null;
    BufferedWriter out = null;
    BufferedReader in = null;
    BufferedReader stdIn = null;

    public clientbai1(String host, int port){
        try{
            socket = new Socket(host, port);
            System.out.println("Client connected ");
            System.out.println("Nhập MSSV: ");
            out = new OutputStreamWriter(socket.getOutputStream());
            in = new InputStreamReader(socket.getInputStream());
            stdIn = new BufferedReader(new InputStreamReader(System.in));
            String line="";
            while(!line.equals("bye")){
                try{
                    //nhập mssv cần tra cứu
                    line = stdIn.readLine();
                    System.out.println("Client sent");
                    System.out.println("MSSV: " + line);
                    out.write(line);
                    out.newLine();
                    out.flush(); // đẩy dữ liệu qua server

                    //nhận kết quả từ server
                    String da= in.readLine();
                    System.out.println("Kết quả: " +da);
                    System.out.println("-----");
                }catch(Exception e){
                    System.err.println(e);
                }
            }
            in.close();
            out.close();
            stdIn.close();
            socket.close();
        }catch(Exception e){
            System.out.println(e);
        }
    }

    public static void main(String[] args){
        clientbai1 client = new clientbai1("localhost",6000);
    }
}
```

* Server

```
import javax.xml.xpath.XPath;
import javax.xml.xpath.XPathFactory;
import java.io.*;
import java.lang.System.*;
import java.net.*;
import java.util.*;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.xml.xpath.XPath;
import org.jsoup.Connection;
import org.jsoup.Jsoup;
import org.jsoup.nodes.*;
import org.jsoup.select.Elements;
public class serverbai1{
    String res ;
    private Socket socket = null;
    private ServerSocket serverSocket = null;
    BufferedReader in = null;
    BufferedWriter out = null;

    public serverbai1(int port) {
        try {
            System.out.println("Server is ready ");
            serverSocket = new ServerSocket(port);
            System.out.println("Server started !");
            System.out.println("Waiting for a client ...");
            socket = serverSocket.accept();
            System.out.println("Client accepted !");

            in = new BufferedReader(new InputStreamReader(
                socket.getInputStream()));
            out = new BufferedWriter(new OutputStreamWriter(
                socket.getOutputStream()));

            String linereceived = "";
            while (!linereceived.equals("bye")) {
                linereceived = in.readLine();
                System.out.println("Received from a client: " + linereceived);
                String linesent = "";
                if (linereceived.equals("bye")) {
                    linesent = "Đừng";
                } else {
                    final long startTime =System.currentTimeMillis();
                    String url= "http://thongtindaotao.sgu.edu.vn/Default.aspx?page=xemdiemthi&id="+linereceived;
                    String domain ="http://thongtindaotao.sgu.edu.vn/";
                    try{
                        String urlLog =domain +"default.aspx?page=nhapmasv&flag=XemDiemThi";
                        String userAgent="Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.45 Safari/537.36";
                        Connection.Response response = Jsoup.connect(urlLog)
                            .method(Connection.Method.GET)
                            .execute();
                        Document login = response.parse();
                    }
                }
            }
        }
    }
}
```

```

//Lấy thông tin sinh viên và điểm học kỳ gần nhất
response = Jsoup.connect(urlLog)
                .data("__EVENTTARGET","");
                .data("__EVENTARGUMENT","");
                .data("__VIEWSTATE", login.getElementById("__VIEWSTATE").val())
                .data("__VIEWSTATEGENERATOR", login.getElementById("__VIEWSTATEGENERATOR").val())
                .data("ctl100$ContentPlaceHolder1$ctl100$txtMaSV",linereceived)
                .data("ctl100$ContentPlaceHolder1$ctl100$btnOK","OK")
                .userAgent(userAgent)
                .timeout(0)
                .followRedirects(true)
                .cookies(response.cookies())
                .method(Connection.Method.POST)
                .execute();

login=response.parse();
String maSV= login.getElementById("ctl100_ContentPlaceHolder1_ctl100_ucThongTinSV_lblMaSinhVien").text();
String hoTen= login.getElementById("ctl100_ContentPlaceHolder1_ctl100_ucThongTinSV_lblTenSinhVien").text();
String noiSinh= login.getElementById("ctl100_ContentPlaceHolder1_ctl100_ucThongTinSV_lblNoiSinh").text();
String nganh= login.getElementById("ctl100_ContentPlaceHolder1_ctl100_ucThongTinSV_lbNganh").text();
String khoa= login.getElementById("ctl100_ContentPlaceHolder1_ctl100_ucThongTinSV_lblKhoa").text();
System.out.println(maSV+ " - "+hoTen+ " - "+noiSinh+ " - "+nganh+ " - "+khoa);
linesent = maSV+ " - "+hoTen+ " - "+noiSinh+ " - "+nganh+ " - "+khoa;
//Lấy điểm học kỳ bắt đầu
XPath xpath = XPathFactory.newInstance().newXPath();
String url1= domain +"Default.aspx?page=xemdiemthi&id="+linereceived;
response = Jsoup.connect(url1)
                .method(Connection.Method.GET)
                .execute();
response = Jsoup.connect(url1)
                .data("__EVENTTARGET","");
                .data("__EVENTARGUMENT","");
                .data("__VIEWSTATE",login.getElementById("__VIEWSTATE").val())
                .data("__VIEWSTATEGENERATOR",login.getElementById("__VIEWSTATEGENERATOR").val())
                .data("ctl100$ContentPlaceHolder1$ctl100$txtChonHK", "20191")
                .data("ctl100$ContentPlaceHolder1$ctl100$btnChonHK","Xem")
                .userAgent(userAgent)
                .timeout(0)
                .followRedirects(true)
                .cookies(response.cookies())
                .method(Connection.Method.POST)
                .execute();

Document doc = response.parse();
Elements diemTK = doc.selectXPath("//*[@id=\"ctl100_ContentPlaceHolder1_ctl100_div1\"]/table/tbody/tr[9]/td/span[2]");
res = diemTK.text();
/*for(Element ex : diemTK)
    System.out.println(ex.text());
res = new String(diemTK.size());
for(int i = 0 ; i < diemTK.size(); i++){
    res[i] = diemTK.get(i).text();
    System.out.println(" " + res[i]);
}
*/
} catch (IOException e) {
    System.out.println(e);
}
}
}
System.out.println("Send to a client: " + linesent+ "; Điểm: " +res);
out.write(linesent+"; Điểm: " + res);
out.newLine();
out.flush();
System.out.println("-----");
}
in.close();
out.close();
socket.close();
} catch (IOException ex) {
    ex.printStackTrace();
}
}
public static void main(String[] args) {
    serverbai1 server= new serverbai1(6000);
}
}

```

17. Viết chương trình gửi tin nhắn hai chiều (chuỗi đảo ngược) giữa client-server sử dụng UDP socket.

- Client gửi 1 chuỗi ký tự bất kỳ đến server
- Server nhận và gửi chuỗi đảo ngược từng từ về client.
- Client xuất kết quả ra console, chương trình kết thúc khi client gửi chuỗi bye.

VD: client gửi chuỗi hello world, server trả lại chuỗi olleh dlrow

***Client:**

```

import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.net.SocketException;
import java.net.UnknownHostException;
import java.util.Scanner;
import java.util.logging.Level;
import java.util.logging.Logger;
|
public class client {

    public static int destPort =1234;
    public static String hostname= "localhost";
    public static void main(String[] args) throws SocketException, IOException {
        try {
            DatagramSocket socket;
            DatagramPacket send, receive;
            InetAddress add;
            Scanner stdIn;
            add= InetAddress.getByName(hostname);
            socket =new DatagramSocket();
            stdIn=new Scanner(System.in);
            while(true){
                System.out.print("Client input: ");
                String tmp =stdIn.nextLine();
                byte[] data=tmp.getBytes();
                send= new DatagramPacket(data,data.length,add,destPort);
                System.out.println("Client sent "+tmp+" to "+add.getHostAddress()+" form port "+socket.getLocalPort());
                socket.send(send);
                if(tmp.equals("bye")){
                    System.out.println("Client socket closed");
                    stdIn.close();
                    socket.close();
                    break;
                }
                receive =new DatagramPacket(new byte[9999],9999);
                socket.receive(receive);
                tmp=new String (receive.getData(),0,receive.getLength());
                System.out.println("Client get: "+tmp+" from server");
            }
        } catch (UnknownHostException ex) {
            Logger.getLogger(client.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
}

```

* Server:

```

import java.util.logging.Level;
import java.util.logging.Logger;

import java.io.File;
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.SocketException;
import java.net.UnknownHostException;
import java.util.HashMap;
import java.util.Map;
import java.util.Scanner;
import java.util.StringTokenizer;
import java.util.logging.Level;
import java.util.logging.Logger;

|
public class server {
    public static int bufsize =9999;
    public static int port =1234;

    public static void main(String[] args) throws SocketException, IOException {
        DatagramSocket socket;
        DatagramPacket send, receive;
        String s = null;

        try{
            socket=new DatagramSocket(port);
            receive =new DatagramPacket(new byte[bufsize],bufsize);

            while(true){
                socket.receive(receive);
                String tmp=new String (receive.getData(),0,receive.getLength());
                System.out.println("Server receive: "+tmp+" from "+receive.getAddress().getHostAddress()+" at port "+socket.getLocalPort());
                if(tmp.equals("bye")){
                    System.out.println("Server socket closed");
                    socket.close();
                    break;
                }
                StringBuilder newline = new StringBuilder();
                newline.append(tmp);
                s = newline.reverse().toString();
                send=new DatagramPacket(s.getBytes(),s.getBytes().length,receive.getAddress(),receive.getPort());
                System.out.println("Server sent back "+s+" to client");
                socket.send(send);
            }
        } catch (UnknownHostException ex) {
            Logger.getLogger(client.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
}

```