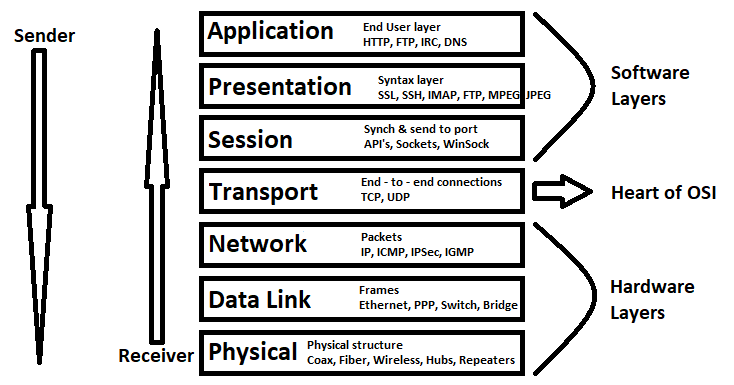
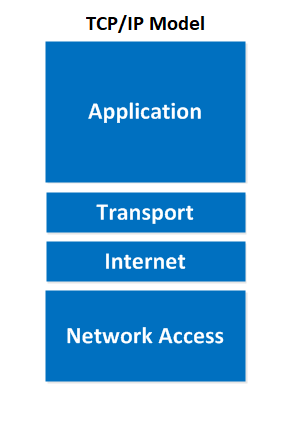
# **I Lý Thuyết:**

1. **Mạng máy tính:**
   1. Là một tập hợp các máy tính và các thiết bị được kết nối với nhau theo một cấu trúc nào đó và qua đó chúng trao đổi thông tin, chia sẻ tài nguyên qua lại cho nhau.
   2. Các máy tính được kết nối với nhauc ó thể trong cùng một phòng, một tòa nhà, một thành phố hoặc trên phạm vi toàn cầu.
   3. Mạng máy tính bao gồm 3 thành phần chính:
      1. Các máy tính.
      2. Các thiết bị mạng đảm bảo kết nối các máy tính với nhau.
      3. Phần mềm cho phép thực hiện việc trao đổi thông tin giữa các máy tính.
2. **Phân loại mạng:**
   1. **Khoảng cách địa lý:**
      1. Mạng cục bộ (LAN).
      2. Mạng đô thị (MAN).
      3. Mạng diện rộng (WAN).
      4. Mạng toàn cầu (GAN).
   2. **Phương tiện kết nối (môi trường truyền thông):**
      1. Mạng có dây.
      2. Mạng không dây.
   3. **Mô hình mạng:**
      1. Mô hình khách – chủ (client-server).
      2. Mô hình ngang hàng (peer-to-peer).
      3. Mô hình dựa trên nền web.
3. **Thông số mạng:**
   1. **Băng thông (Bandwidth - B):**
      1. Là một trong những đặc trưng cơ bản trong môi trường truyền dẫn.
      2. Là đại lượng được dùng để chỉ một khối lượng dữ liệu có thể truyền tải được trong một thời gian nhất định.
      3. Đối với các thiết bị kỹ thuật số băng thông được tính với đơn vị dps (bit mỗi giây) hay Bps (byte mỗi giây).
   2. **Độ trễ (Latency - L) :**
      1. là khoảng thời gian chuyển một thông điệp từ nút này đến nút khác trong hệ thống mạng.
   3. **Thông lượng (Throughput - T):**
      1. là lượng dữ liệu đi qua đường truyền trong một đơn vị thời gian, hay là băng thông thực sự mà các ứng dụng mạng được sử dụng trong một thời gian cụ thể.
      2. Thông lượng nhỏ hơn nhiều so với băng thông tối đa có thể có củ môi trường truyền dẫn được sử dụng.
      3. Phụ thuộc vào các yếu tố như: cách liên kết, môi trường truyền dẫn, cộng nghệ, dữ liệu truyền,...
4. **Các phương pháp truyền thông dữ liệu:**
   1. Unicast: một nút nguồn muốn gửi một thông điệp đến duy nhất một nút đích trên hệ thống mạng.
   2. Multicast: một nút nguồn muốn gửi một thông điệp đến một nhóm các nút đích trên hệ thống mạng.
   3. Broadcast: một nút nguồn muốn gửi một thông điệp đến tất cả các nút đích khác trên hệ thống.
5. **Chi tiết các mô hình:**
   1. **Mạng LAN:**
      1. Là một nhóm các máy tính và thiết bị truyền thông mạng được kết nối với nhau trong một khu vực nhỏ như: tòa nhà, trường đại học, khu giải trí.
      2. Đặc điểm:
         1. Băng thông lớn, ít lỗi có khả năng chạy các ứng dụng trực tuyến như xem phim, giải trí, hội thảo qua mạng.
         2. Kích thước mạng bị giới hạn bởi thiết bị.
         3. Chí phí thiết kế, lắp đặt rẻ.
         4. Quản trị đơn giản.
   2. **Mạng MAN:**
      1. Gần giống như mạng LAN nhưng giới hạn kích thước của nó là một thành phố hay một quốc gia.
      2. Mạng MAN kết nối các mạng LAN lại với nhau thông qua môi trường truyền dẫn và các phương thức truyền thông khác nhau.
      3. Đặc điểm:
         1. Băng thông ở mức trung bình, nhiều lỗi đủ để phục vụ các ứng dụng cấp thành phố hay quốc gia như chính phủ điện tử, thương mại điện tử, các ứng dụng của các ngân hàng.
         2. Do kết nối nhiều mạng LAN nên việc quản trị sẻ gặp khó khăn hơn đồng thời độ phức tạo cũng tăng theo.
         3. Chi phí tương đối đắt.
   3. **Mạng WAN:**
      1. Là tập hợp nhiều mạng LAN và MAN được nối lại với nhau thông qua các phương tiện truyền dẫn.
      2. Có phạn vi bao phủ một vùng rộng lớp, có thể là quốc gia, lục địa hay toàn cầu.
      3. Thường lạ mạng của các công ty đã quốc gia hay toàn cầu. Mạng WAN lớn nhất hiện nay là mạng Internet.
      4. Đặc điểm:
         1. Băng thông thấp, dễ mất kết nối, tường chỉ phù hợp với các ứng dụng online như e-mail, ftp, web...
         2. Phạm vi hoạt động không giới hạn.
         3. Do kết nối nhiều mạng LAN và MAN vơi snhau nên mạng rất phức tạp và các tổ chức toàn cầu phải đứng ra quy định và quản lý.
         4. Chi phí cho các thiết bị và công nghệ mạng WAN rất đắt.
   4. **Mạng GAN:**
      1. Là mạng trải rộng trong nhiều quốc gia, phụ vụ phát triển kinh tế xã hội cho nhưng côn ty siêu quốc gia hoặc nhóm các quốc gia, đường truyền có thể sử dụng cơ sở hạng tầng của viễn thông, mạng Internet là một mạng GAN.
   5. **Ngang hàng (peer – to – peer):**
      1. Trong mô hình này tất cả các máy tính tham gia có vai trò giống nhau.
      2. Mỗi máy tính có thể cung cấp trực tiếp tài nguyên của mình cho máy tính khác vừa có thể sử dụng tài nguyên của máy tính khác trong mạng.
      3. Chỉ thích hợp với mạng có quy mô nhỏ, tài nguyên được quản lý phân tán, chế độ bảo mật kém, xây dựng và bảo trì đơn giản.
   6. **Chủ - khách (client – server):**
      1. Trong mô hình này một vài máy tính được chọn để đảm nhận việc quản lý và cung cấp tài nguyên được gọi là máy chủ (server).
      2. Các máy khác sử dụng tài nguyên này được gọi là máy khách (client).
      3. Máy chủ đảm bảo việc phục vụ các máy khách bằng cách điều khiển việc phân phối tài nguyên nằm trong mạng với mục đích sử dụng chung.
      4. Máy khách sử dụng tài nguyên do máy chủ cung cấp.
      5. Mô hình có ưu điểm là dữ liệu được quản lý tập trung, bảo mật tốt, thích hợp với các mạng trung bình và lớn.
   7. **Dựa trên nền web:**
      1. Do sự phát triển của Internet nên có rất nhiều công ty và cá nhân sử dụng Internet nhưu một mạng “xương sống” và kết nối với mọi người trên toàn cầu.
      2. Mạng trên phạm vi Internet được gọi là mạng liên kết nối và ngày càng trở nên phổ biến.
      3. Người dùng chỉ cần trình duyệt web và một kết nối Internet để chia sẻ các tập tin, tải các ứng dụng, xem video hoặc tham gia học trực tuyến.
   8. **Mạng có dây:**
      1. Cáp truyền thông có thể là cáp xoắn đôi, cáp đồng trục, cáp quang.
      2. Để tham gia vào mạng máy tính cần có vỉ mạng được kế nối với cáp mạng nhờ giắc cắm.
      3. Một số thiết bị kết nối có dây: Hub, Bridge, Switch, Router.
      4. Kiểu bổ trí các máy tính trong mạng: có 3 kiểu cơ bản là đường thẳng, vòng, hình sao.
   9. **Mạng không dây:**
      1. Dùng sóng radio, bức xạ hồng ngoại, truyền thông tin qua vệ tính.
      2. Thiết bị WAP (Wireless Access Point): có chức năng kết nối các máy tính trong mạng và kết nối với mạng có dây.
      3. Máy tính phải có vỉ mạng không dây.
      4. Người ta dùng bộ định tuyến không dây ngoài chức năng như điểm truy cập không dây
6. **Phương tiện truyền thông:**
   1. **Kết nối không dây.**
   2. **Kết nối có dây.**
   3. **Chi tiết ở phần mô hình mạng.**
7. **Internet.**
   1. **Khái niệm:** Internet là mạng máy tính khổng lồ, kết nối hàng triệu máy tính trên khắp thế giới và sử dụng bộ giao thức truyền thông TCP/IP. Internet đảm bảo cho mọi người có khả năng thâm nhập đến nhiều nguồn thông tin thường trực, cung cấp các chỉ dẫn bổ ích, dịch vụ mua bán, truyền tệp, thư điện tử và nhiều khả năng khác.
   2. **Cách kết nối:**
      1. **Sử dụng môđem qua đường điện thoại** 
         1. Cài đặt Modem và kết nối qua đường điện thoại
         2. Ký hợp đồng với nhà cung cấp dịch vụ (ISP) để có quyền truy cập (tên truy cập, mật khẩu)
         3. Ưu điểm: thuận tiện cho người sử dụng.
         4. Nhược điểm: tốc độ đường truyền không cao.
      2. **Sử dụng đường truyền riêng (Leased line)**
         1. Thuê đường truyền riêng kết nối từ ISP tới máy tính.
         2. Thường sử dụng cho mạng LAN của các cơ quan, xí nghiệp,…
         3. Ưu điểm: Tốc độ đường truyền cao; Phù hợp với những nơi có nhu cầu kết nối liên tục và trao đổi thông tin với khối lượng lớn
      3. **Một số phương thức kết nối khác** 
         1. Sử dụng đường truyền ADSL (đường truyền thuê bao số bất đối xứng). Tốc độ truyền dữ liệu cao hơn rất nhiều so với đường điện thoại. Giá thành ngày càng hạ -> ngày càng nhiều ngưởi sử dụng.
         2. Công nghệ không dây: Wi-Fi là phương thức mới nhất, thuận tiện nhất. Kết nối Internet mọi lúc, mọi nơi thông qua ĐTDĐ, laptop,…
         3. Ngoài ra, hiện nay nhiều nhà cung cấp dịch vụ kết nối Internet qua đường truyền hình cáp; Kết nối 3G.
8. **Cách các máy tính trong Internet giao tiếp với nhau:**
   1. Các máy tính trong mạng Internet hoạt động và giao tiếp với nhau được là do chúng cùng sử dụng bộ giao thức truyền thông TCP/IP.
   2. Bộ giao thức TCP/IP là tập hợp các quy định về khuôn dạng dữ liệu và phương thức truyền dữ liệu giữa các thiết bị trên mạng.
   3. Bộ giao thức này cho phép hai thiết bị truyền thông trong mạng kết nối với nhau và trao đổi các dòng dữ liệu đã đóng gói hoặc thông tin cần truyền, đảm bảo việc phân chia dữ liệu ở máy gửi thành các gói tin nhỏ hơn có khuôn dạng và kích thước xác định.
   4. Các gói tin được đánh số để máy nhận có thể tập hợ chúng lại một cách đúng đắn như các gói tin ở máy gửi. Khi truyền đi nếu có lỗi không thể khắc phục được thì gói tin sẻ bị truyền trở lại.
9. **Bộ giao thức TCP/IP (chi tiết cho câu 8):**
   1. TCP/IP:
      1. là bộ giao thức truyền thông chung giữa các máy tính trong Internet. Tập hợp các quy định về khuôn dạng dữ liệu và phương thức truyền dữ liệu giữa các thiết bị trong mạng
      2. Gồm nhiều giao thức khác nhau, trong đó 2 giao thức chính là giao thức TCP và giao thức IP.
   2. Giao thức TCP:
      1. Chia các dòng dữ liệu (đã đóng gói) hoặc thông tin cần truyền thành các gói tin nhỏ hơn sau đó phục hồi lại ở máy nhận.
      2. Các gói tin được đánh số xác định nhằm tổng hợp một cách đúng đắn ở máy nhận.
   3. Giao thức IP
      1. Đánh địa chỉ máy gởi và nhận vào gói tin. Để đảm bảo gói tin đến đúng máy nhận.
      2. Khi truyền tin nếu có lỗi không khắc phục được, gói tin sẽ được truyền lại.
   4. Khái niệm địa chỉ IP:
      1. Là địa chỉ dùng để xác định vùng duy nhất của mạng, một máy tính hay một đối tượng cụ thể nào đó trên mạng giúp cho các thiết bị nhận diện và trao đổi dữ liệu với nhau.
      2. Cần có địa chỉ IP để phân biệt các đối tượng trên mạng; hỗ trợ việc tìm kiếm và truy cập thông tin.
      3. Có 2 dạng biểu diễn địa chỉ IP: dạng số và dạng kí tự.
   5. Dịa chỉ IP dạng số:
      1. Ipv4 sử dụng 32bit để mã hóa dữ liệu.
      2. Ipv6 sử dụng 128bit để mã hóa dữ liệu, nó cho phép sử dụng nhiều địa chỉ hơn so với Ipv4.
   6. Địa chỉ IP dạng kí tự - tên miền:
      1. Là địa chỉ được chuyển từ địa chỉ IP dạng số sang dạng kí tự bởi một số máy chủ DNS.
      2. Dễ nhớ và thuận tiện cho người dùng hơn địa chỉ IP dạng số.
      3. Gồm nhiều trường, được phân cách với nhau bởi dấu ‘.’
      4. Ví dụ: tuoitre.vn, laodong.com.vn
      5. Trường cuối thường là viết tắt của tên nước .vn (Việt Nam), .jp (Nhật bản).
   7. IP tĩnh:
      1. Ip tĩnh hay còn gọi là static IP đây là địa chỉ IP cố định không thay đổi.
   8. IP động:
      1. Ip động hay còn gọi là Dynamic IP đây là địa chỉ IP thay đổi liên tục và không cố định.
10. **Các yếu tố cần quan tâm khi thiết kế mạng.**
    1. Số lượng máy tính tham gia mạng.
    2. Tốc độ truyền thông trong mạng.
    3. Địa điểm lắp đặt mạng.
    4. Khả năng tài chính.
11. **Giao thức (Protocol):**
    1. Giao thức truyền thông là bộ các quy tắc phải tuân thủ trong việc trao đổi thông tin trong mạng giữa các thiết bị nhận và truyền dữ liệu.
    2. Giao thức được dùng phổ biến trong các mạng, đặc biệt là mạng toàn cầu Internet là giao thức TCP/IP (Transmisson Control Protocol/Internet Protocol)
12. **Mô hình OSI.**

****

* 1. **Physical:**
     1. Chuyển bit thô qua phương tiện truyền dẫn.
     2. Bits.
     3. Hub, Repeater, Cables.
  2. **Data Link:**
     1. Đảm bảo dữ liệu truyền tin cậy từ 1 node -> khác qua kênh truyền con: Logical Link Control (LLC) & Media Access Control (MAC).
     2. Frames.
     3. NIC, Switch, Bridge.
     4. Functions:
        1. Framing.
        2. Physical addressing.
        3. Error control.
        4. Flow Control.
        5. Access control.
  3. **Network:**
     1. Truyền dữ liệu giữa các host ở các mạng khác nhau (mạng logic), packet routing.
     2. Packet.
     3. Router.
     4. Functions:
        1. Routing.
        2. Logical Addressing.
  4. **Transport:**
     1. Vận chuyển từ đầu đến cuối.
     2. Phân đoạn.
     3. Tầng vận chuyển được gọi là Trái tim của mô hình OSI.
     4. Functions:
        1. Segmentation and Reassembly.
        2. Service Point Addressing.
  5. **Session:**
     1. Thiết lập kết nối, duy trì session.
     2. Data.
     3. Functions:
        1. Session establishment, maintenance and termination.
        2. Synchronization.
  6. **Presentation (Translation):**
     1. Xác lập dạng thức dữ liệu được trao đổi.
     2. Data.
     3. Functions:
        1. Translation (ex ASCII to EBCDIC).
        2. Encryption/ Decryption.
        3. Compression.
  7. **Application:**
     1. Giao diện giữa các chương trình ứng dụng của người dùng và mạng. Cung cấp các dịch vụ mạng.
     2. Data.
     3. Functions:
        1. Network Virtual Terminal.
        2. FTAM-File transfer access and management.
        3. Mail Services.
        4. Directory Services.

1. **Giao thức:**
   1. **Giao thức truyền thông:**
      1. Giao thức truyền thông là bộ các quy tắc phải tuân thủ trong việc trao đổi thông tin trong mạng giữa các thiết bị nhận và truyền dữ liệu.
   2. **Giao thức TCP/IP.**
      1. Bộ giao thức TCP/IP, ([tiếng Anh](https://vi.wikipedia.org/wiki/Ti%E1%BA%BFng_Anh): Internet protocol suite hoặc IP suite hoặc TCP/IP protocol suite - bộ giao thức liên mạng), là một bộ các [giao thức truyền thông](https://vi.wikipedia.org/wiki/Giao_th%E1%BB%A9c_truy%E1%BB%81n_th%C3%B4ng) cài đặt [chồng giao thức](https://vi.wikipedia.org/wiki/Ch%E1%BB%93ng_giao_th%E1%BB%A9c) (protocol stack) mà [Internet](https://vi.wikipedia.org/wiki/Internet) và hầu hết các mạng máy tính thương mại đang chạy trên đó. Bộ giao thức này được đặt tên theo hai giao thức chính của nó là [TCP](https://vi.wikipedia.org/wiki/TCP) (Giao thức Điều khiển Giao vận) và [IP](https://vi.wikipedia.org/wiki/IP) (Giao thức Liên mạng). Chúng cũng là hai giao thức đầu tiên được định nghĩa.
      2. Như nhiều bộ giao thức khác, bộ giao thức TCP/IP có thể được coi là một tập hợp các tầng, mỗi tầng giải quyết một tập các vấn đề có liên quan đến việc truyền dữ liệu, và cung cấp cho các [giao thức tầng cấp trên](https://vi.wikipedia.org/w/index.php?title=Giao_th%E1%BB%A9c_t%E1%BA%A7ng_c%E1%BA%A5p_tr%C3%AAn&action=edit&redlink=1) một dịch vụ được định nghĩa rõ ràng dựa trên việc sử dụng các dịch vụ của các tầng thấp hơn. Về mặt logic, các tầng trên gần với người dùng hơn và làm việc với dữ liệu trừu tượng hơn, chúng dựa vào các giao thức tầng cấp dưới để biến đổi dữ liệu thành các dạng mà cuối cùng có thể được truyền đi một cách vật lý.
2. **Mô hình TCP/IP:**
   1. Mô hình TCP/IP:
      1. TCP/IP là viết tắt của cụm từ Transmission Control Protocol/Internet Protocol, là một tập hợp các gioa thức trao đổi thông tin được sử dụng để truyền tải và kết nối các thiết bị trong mạng Internet. Cụ thể hơn, TCP/IP chỉ rõ cho chúng ta cách thức đóng gói thông tin (còn được gọi là gói tin), được gửi và nhận bởi các máy tính có kết nối với nhau.



* + 1. Mô hình TCP/IP tiêu chuẩn bao gồm 4 tầng được chồng lên nhau là:
       1. Network Access - tầng vật lý.
       2. Internet – tầng mạng.
       3. Transport – tầng giao vận.
       4. Application – tầng ứng dụng.
    2. Tầng ứng dụng (Application):
       1. Nó cung cấp giao tiếp đến người dùng.
       2. Cung cấp các ứng dụng cho phép người dùng trao đổi dữ liệu ứng dụng thông qua các dịch vụ mạng khác nhau (như duyệt web, chat, gửi mail,...).
       3. Dữ liệu khi đến đây sẽ được định dạng theo kiểu byte nối byte, cúng với đó là các thông tin định tuyến giúp xác định đường đúng của một gói tin.
       4. Một số giao thức trao đổi dữ liệu: FTP, TFTP, SMTP, Telnet, SNMP, DNS.
    3. Tầng giao vận (Transport):
       1. Chịu trách nhiệm duy trì liên lạc đầu cuối trên toàn mạng.
       2. Tầng này có 2 giao thức chính là TCP (Transmisson Control Protocol) và UDP (User Datagram Protocol).
          - TCP sẻ đảm bảo chất lượng truyền gửi gói tin, nhưng tốn khá nhiều thời gian để kiểm tra đầy đủ thông tin từ thứ tự dữ liệu cho đến việc kiểm soát vấn đề tắc nghẽn lưu lượng dữ liệu.
          - Trái với TCP, UDP có thấy tốc độ truyền tải nhanh hơn nhưng hơn nhưng lại không đảm bảo được chất lượng dữ liệu được gửi đi (nó không quan tâm dữ liệu có đến được đích hay không).
    4. Tầng mạng (Internet):
       1. Xử lý quá trình truyền gói tin trên mạng.
       2. Định tuyến: tìm tuyến đường qua các nút trung gian để gửi dữ liệu từ nguồn tới đích.
       3. Chuyển tiếp: chuyển tiếp gói tin từ cổng nguồn tới cổng đích theo tuyến đường.
       4. Định địa chỉ: định danh cho các nút mạng.
       5. Đóng gói dữ liệu: nhận dữ liệu từ giao thức ở trên, chèn thêm phần header chứa thông itn của tầng mạng và tiếp tục được chuyển đến tầng tiếp theo.
       6. Đảm bảo chất lượng dịch vụ: đảm bảo các thông số phù hợp vủa đừng truyền theo từng dịch vụ.
       7. Giao thức của tầng này bao gồm IP, ICMP, IGMP.
    5. Tầng vật lý (Network Access):
       1. Nó là sự kết hợp của tầng Data Link và Physical trong mô hình OSI.
       2. Là tầng thấp nhất trong mô hình TCP/IP.
       3. Chịu trách nhiệm truyền dữ liệu giữa các thiết bị trong cùng một mạng. Tại đây các gói dữ liệu được đóng vào khung và được định tuyến đi đến đích được chỉ định ban đầu.
  1. Các thức hoạt động:
     1. Khi truyền dữ liệu, quá trình tiến hành từ tầng trên xuống tầng dưới qua mỗi tầng dữ liệu được thêm vào thông tin điều khiển gọi là Header. Khi nhận được dữ liệu thì quá trình này xảy ra ngược lại, dữ liệu được truyền từ dưới lên và qua mỗi tầng thì phần header tương ứng sẽ được lấy đi và khi đến tầng trên cùng thì dữ liệu không còn phần Header nữa.
        1. Ơ đây, Ip có vai trò quan rọng, nó cho phép các gói tin được gửi đến đích đã định sẵn, bằng cách thêm các thông tin dẫn đường (Header) vào các gói tin để các gói tin được đến đúng đích như đã định sẵn ban đầu.
        2. Giao thức TCP đóng vai trò kiểm tra và đảm bảo sự an toàn cho mỗi gói tin đi qua mỗi trạm. Trong quá trình này, nếu giao thức TCP nhận thấy gói tin bị lỗi, một tín hiệu sẽ được truyền đi và yêu cầu hệ thống gửi lại một gói tin khác.
     2. Dữ liệu qua các tầng: mỗi tầng khác nhau dữ liệu truyền là khác nhau.
        1. Tầng ứng dụng: dữ liệ là các luồng gọi là stream.
        2. Tầng giao vận: đơn vị dữ liệu mà TCP gửi xuống gọi là TCP segment.
        3. Tầng mạng: dữ liệu mà IP gửi xuống tầng dưới gọi là IP   
           Datagram.
        4. Tầng liên kết: dữ liệu được truyền đi gọi là frame.
  2. Ưu điểm của mô hình TCP/IP:
     1. Không chịu sự kiểm soát của bất kỳ tổ chức nào nên có thể tự do trong việc sử dụng.
     2. Có khả năng tương thích cao với tất cả các hệ điều hành, phần cứng máy tính và mạng nên hoạt động hiệu quả với nhiều hệ thống khác nhau.
     3. Có khản năng mở rộng cao, có thể định tuyến nên có thể xác định được đường dẫn hiệu quả nhất.

1. **Một số giao thức trao đổi dữ liệu:**
   1. FTP (File Transfer Protocol): giao thức chạy trên nền TCP cho phép truyền các file ASCII hoặc nhị phân theo 2 chiều.
   2. TFTP (Trival File Transfer Protocol): giao thức truyền file chạy trên nền UDP.
   3. SMTP (Simple Mail Transfer Protocol): giao thức dùng để phân phối thư điện tử.
   4. Telnet: cho phép truy nhập từ xa để cấu hình thiết bị.
   5. SNMP (Simple Network Managerment Protocol): là ứng dụng chạy trên nền UDP, cho phép quản lý và giám sát các thiết bị mạng từ xa.
   6. DNS (Domain Name System): là giao thức phân giải tên miền, được sử dụng trong hỗ trợ truy nhập Internet.
2. **Mô hình đa tầng.**
3. **Trình bày câu 1 nhỏ câu 2:**
4. **Socket.**
   1. **Khái niệm:**
   * 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.

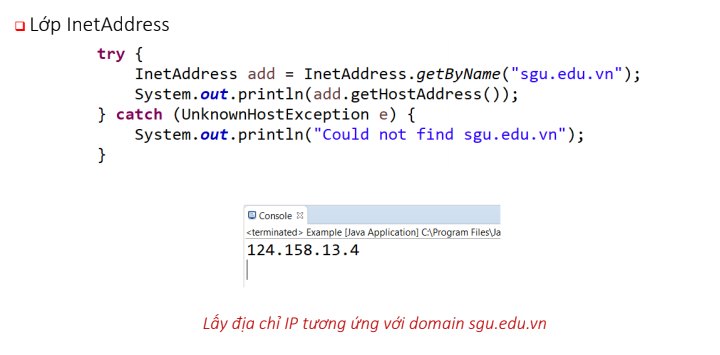
Socket = IP + Port

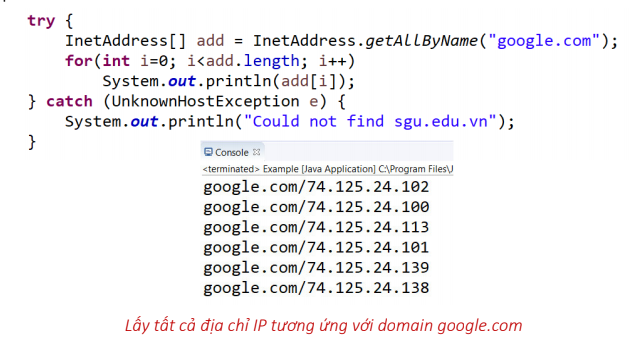
=> Địa chỉ socket xác định một đầu mút cuối truyền thông. Nó chỉ ra tiến trình truyền thông nào(port) và chạy trên trên máy nào(IP) sẽ thực hiện truyền thông.

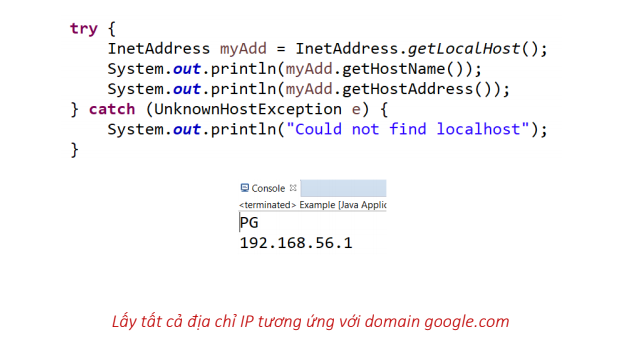
* 1. **Phân loại:**
* Stream socket
* Datagram socket
* Raw socket: cho phép truyền thông đến các giao thức ở tầng mạng thấp hơn cả tầng transport (VD: giao thức ICMP của tầng Internet)
  1. **Lập trình với địa chỉ IP:**

#### **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 InetAddess InetAddress.getByName(String hostname)
* public static InetAddess [] InetAddress.getAllByName(String hostname)
* public static InetAddess InetAddress.getLocalHost()



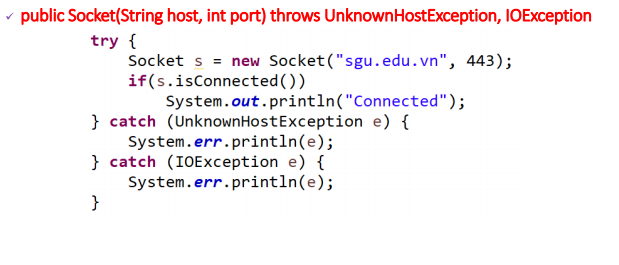




1. **TCPSocket.**
   1. **Chương trình phía Client:**

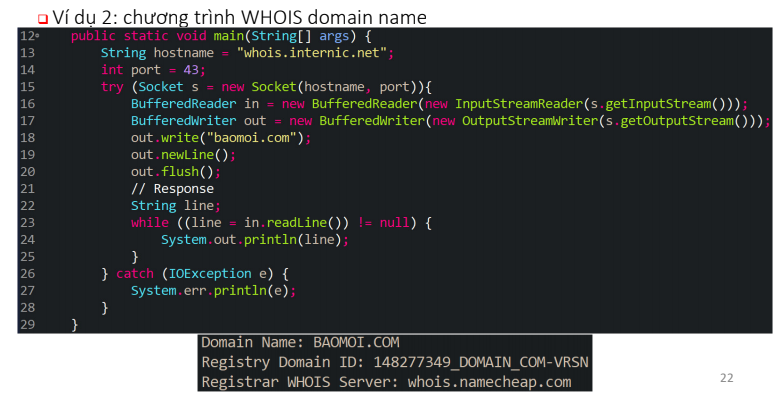
**Lớp Socket: chương trình phía client**

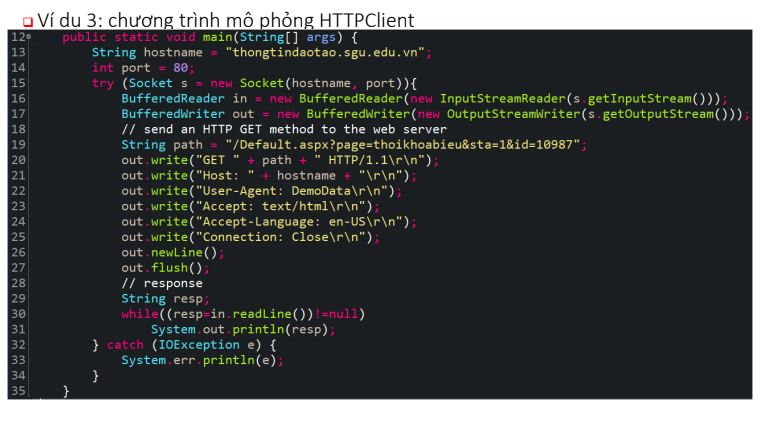
* Dùng để tạo đối tượng socket cho phép truyền thông với giao thức TCP hoặc UDP. (giao thức UDP thường sử dụng lớp DatagramSocket thay vì lớp Socket)
* Constructor:

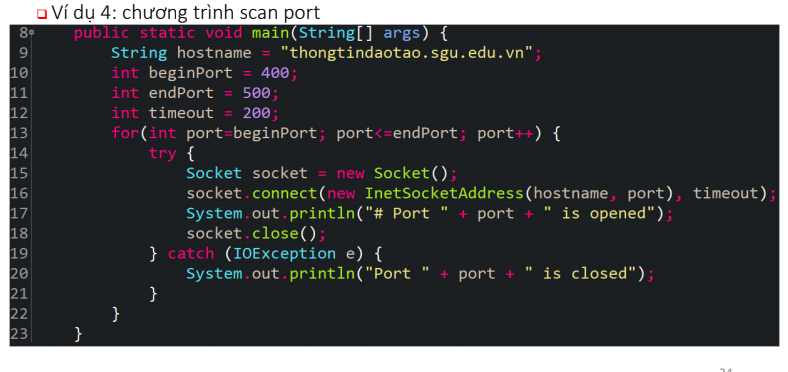


* public Socket(String host, int port) throws UnknownHostException, IOException
* public Socket(InetAddress host, int port) throws IOException
* public Socket(String host, int port, InetAddress interface, int localPort) throws
* IOException, UnknownHostException
* public Socket(InetAddress host, int port, InetAddress interface, int localPort) throws IOException
* public InetAddress getInetAddress( ): trả về địa chỉ mà socket kết nối đến
* public int getPort( ): port number trên máy trạm từ xa đang kết nối vớisocket.
* 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





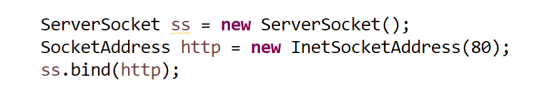




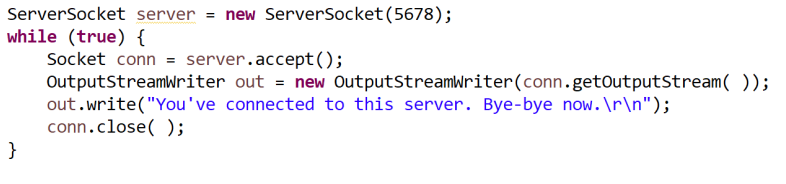
* 1. **Chương trình phía Server:**

**Lớp ServerSocket: chương trình phía server**

* 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().

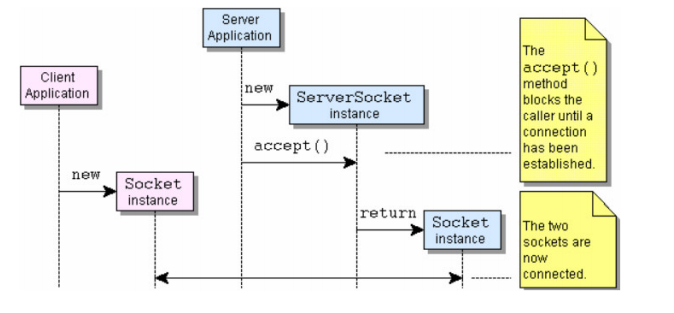


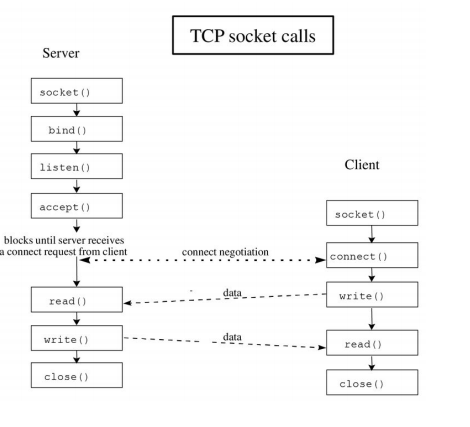
* Phương thức accept()
* public Socket accept( ) throws IOException
* Đặt đối tượng ServerSocket ở trạng thái “nghe” tại số cổng xác định chờ tín hiệu kết
* nối gửi đến từ client.
* Khi có tín hiệu kết nối gửi tới phương thức sẽ trả về đối tượng Socket mới để phục vụ
* kết nối đó.
* Khi xảy ra lỗi nhập/xuất,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



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

**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

**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.

**Mô hình gửi dữ liệu qua lại giữa client và server:**

1. **UDPSecket.**

**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)

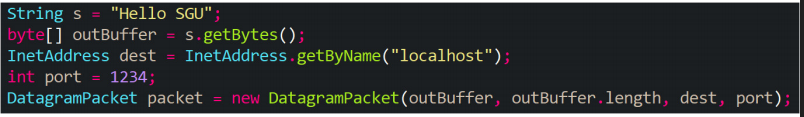
* inBuffer: bộ đệm nhập, chứa dữ liệu gói tin nhận
* length: kích cỡ dữ liệu gói tin nhận, xác định bằng inBuffer.length



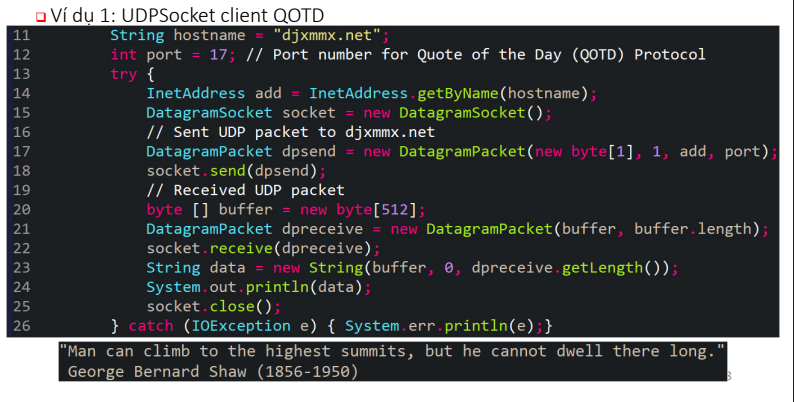
* Constructor tạo gói tin (datagram) gửi:

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)

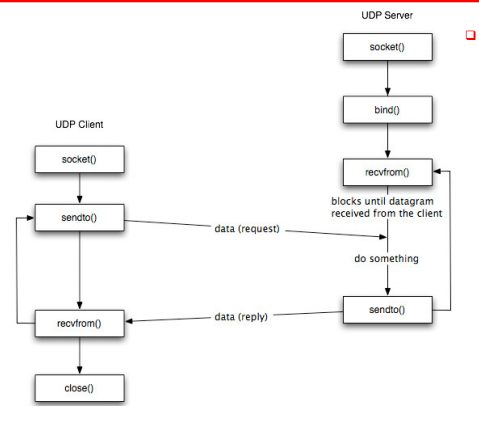


* Các phương thức đối với gói tin nhận:
* public InetAddress getAddress(): trả về đối tượng InetAddress của máy gửi trong gói tin nhận.
* public int getPort(): trả về port number của máy gửi trong gói tin nhận.
* public byte[] getData(): trả về dữ liệu chứa trong gói tin dạng byte
* public int getLength(): 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....
* 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()



**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.



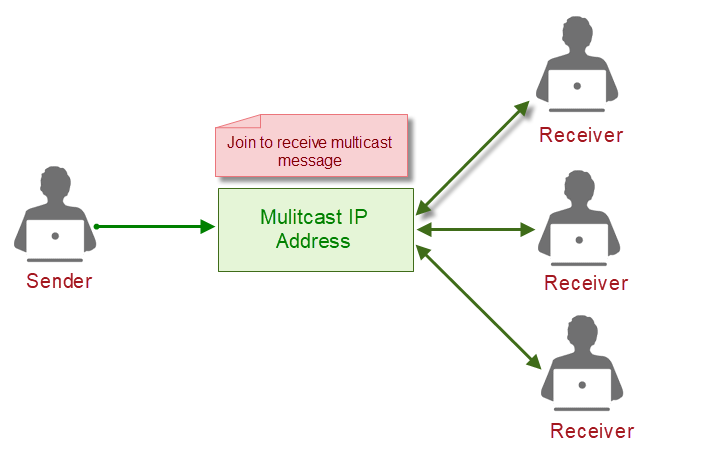
**Lưu ý:**

* 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

1. **Multicast.**
   1. **Multicast là gì:**

* Các cơ chế Socket TCP và UDP đã giới thiệu ở trên đề gọi là unicast, nghĩa là giao tiếp chỉ diễn ra giữa một máy tính gửi và một máy tính nhận.
* Multicast là việc gửi quảng bá (broadcast) nhưng đến một nhóm máy tính ở cùng một địa chỉ cho trước. Địa chỉ multicast là địa chỉ lớp D được xác định trong khoảng 244.0.0.0 đến 239.255.255.255. Địa chỉ 244.0.0.0 là địa chỉ riêng nên không sử dụng được.
* Multicast được sử dụng trong game nhiều người chơi, trong những ứng dụng mà đối tượng là nhiều thiết bị hay nhiều máy tính cùng nhận một loại thông tin. Multicast cũng được sử dụng trong giải thuật vạch đường (Routing Protocol), khi các router muốn cập nhật thông tin với nhau.
* Java hỗ trợ Multicast thông qua lớp java.net.MulticastSocket. Một Multicast Socket là 1 DatagramSocket (UDP) có khả năng gia nhập (joining) vào một nhóm các máy tính multicast trên mạng. Khi một máy tính nào gửi thông điệp đến nhóm thì tất cả các máy tính trong đó đều nhận được.



* 1. **Một số phương thức cần thiết để xây dựng các chương trình Multicast**
* public **MulticastSocket(int port)**: Tạo Socket kiểu Multicast với số hiệu cổng được xác định trong tham số (port).
* public void **joinGroup(InetAddress group)** : Tham gia nhóm Multicast tại địa chỉ xác định.
* public void **leaveGroup(InetAddress group)** : Rời khỏi nhóm Multicast tại địa chỉ xác định.
* public void **send(DatagramPacket dp)** : Dùng để gởi một DatagramPacket đi.  
  public synchronized void **receive(Datagrampacket dp)** : Chờ nhận một DatagramPacket.

1. **Xây dựng ứng dụng mạng phía server.**
   1. **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.
* 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
  1. **Tạo Thread**

Lớp Thread cung cấp các constructor và phương thức để tạo và thực hiện các hoạt động trên một thread. Lớp Thread extends từ lớp Object và implements Runnable interface.

* Các constructor thường được sử dụng của lớp Thread
  + Thread()
  + Thread(String name)
  + Thread(Runnable r)
  + Thread(Runnable r,String name
* Các phương thức thường được sử dụng của lớp Thread
  + **public void run():**Được sử dụng để thực hiện hành động cho một thread..
  + **public void start():**Bắt đầu thực hiện thread. JVM gọi phương thức run() trên thread.
  + **public void sleep(long miliseconds):**Làm cho thread hiện tại tạm ngừng thực thi cho số mili giây quy định.
  + **public void join():**Đợi cho một thread chết.
  + **public void join(long miliseconds):**Đợi cho một thread chết với các mili giây quy định.
  + **public int getPriority():**Trả về mức độ ưu tiên của thread.
  + **public int setPriority(int priority):**Thay đổi mức độ ưu tiên của thread.
  + **public String getName():**Trả về tên của thread.
  + **public void setName(String name):**Thay đổi tên của thread.
  + **public Thread currentThread():**Trả về tham chiếu của thread đang được thi hành.
  + **public int getId():**Trả về id của thread.
  + **public Thread.State getState():**Trả về trạng thái của thread.
  + **public boolean isAlive():**Kiểm tra nếu thread còn sống.
  + **public void yield():**Làm cho các đối tượng thread đang thực thi để tạm thời tạm dừng và cho phép các thread khác để thực thi.
  + **public void suspend():**Được sử dụng để hoãn lại các thread (không dùng nữa).
  + **public void resume():**Được sử dụng để tiếp tục các thread đang bị hoãn (không dùng nữa).
  + **public void stop():**Được sử dụng để dừng thread (không dùng nữa).
  + **public boolean isDaemon():**Kiểm tra nếu thread là một luồng hiểm.
  + **public void setDaemon(boolean b):**Đánh dấu thread là luồng hiểm hoặc luồng người dùng.
  + **public void interrupt():**Ngắt thread.
  + **public boolean isInterrupted():**Kiểm tra nếu thread đã bị ngắt.
  + **public static boolean interrupted():**Kiểm tra nếu thread hiện tại đã bị ngắt.

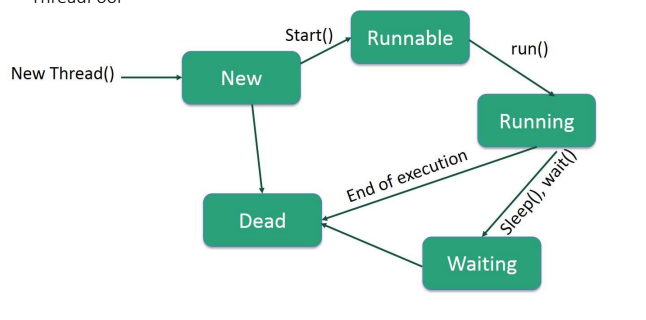




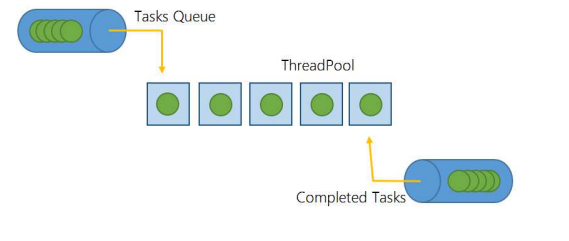
* **Ưu điểm Implement Runable interface:**

Không cần kế thừa từ lớp Thread

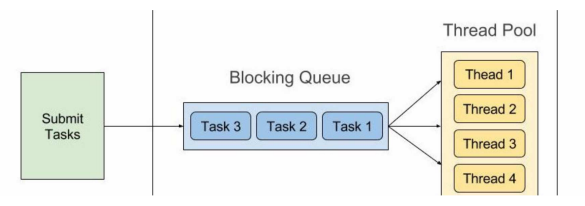
ThreadPool



* 1. **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
  1. **Excutor Framework**

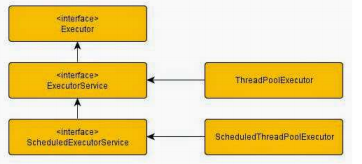
**Excutor là gì:**

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

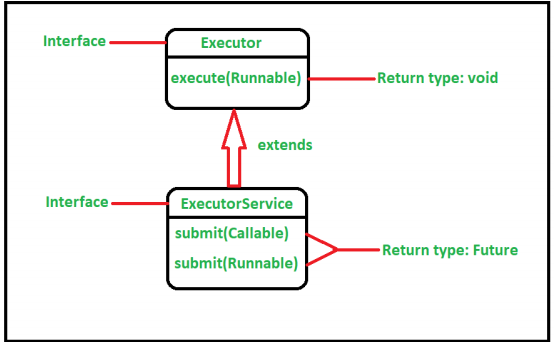
Executor tách riêng các chi tiết của việc tạo Thread, lập lịch (scheduling), ... nhằm tập trung phát triển logic của tác vụ mà không quan tâm đến các chi tiết quản lý Thread.

**Excutor Framework**

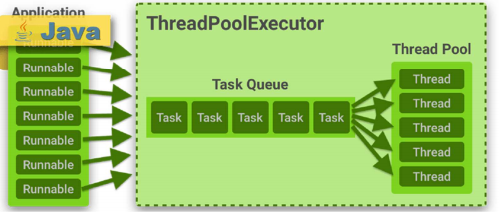
* “Executor framework” trong gói java.util.concurrent giúp tạo và quản lý các “ThreadPool” và “Thread Factories”.
* Executor: là interface cha của tất cả các Executor, method: execute(Runnable)
* ExecutorService: theo dõi tiến trình của các tác vụ trả về giá trị (Callable) thông qua đối
* tượng Future, 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()



* ExecutorService:



* Tạo 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.
  1. **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!



**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.

1. **Lập trình mạng phân tán.**

II Bài Tập

* + - 1. Chuyển ký tự hoa sang thường – thường sang hoa.

public static String hoaThuong\_ThuongHoa(String data) {

String ketQua = "";

for (int i = 0; i < data.length(); i++) {

if (data.charAt(i) >= 'a' && data.charAt(i) <= 'z') {

ketQua += Character.toUpperCase(data.charAt(i));

} else {

if (data.charAt(i) >= 'A' && data.charAt(i) <= 'Z') {

ketQua += Character.toLowerCase(data.charAt(i));

} else {

ketQua += data.charAt(i);

}

}

}

return ketQua;

}