AĞ Programlama

Sunucu

Web pages
FTP server (dosya transferi)

Port

(1024 ~ 65535)

Sunucuya bağlanma adresidir. (65535)
Sunucu bir sürü bağlantı noktasından oluşur
1-1023 arası well known (80 Web port)

Client

Browsers

P2P & WWW

WorldwideWeb → vardır.
→ ikili bağlantı olmaz

Maliyetli P2P → Mesajlaşmalarda kullanılır
↓
Güvenli

Sunucu & Host

Sunucu → Belirli bir port'u dinler
İstemciler port üzerinden talepte bulunur.
Host → Talebi uygun sunucuya aktarır.

Socket

Farklı istemcilerden gelen talepleri ayırırlar

process —o—|—o— process
socket

- İstemci istek oturken socket oluşturur
- Sunucu talep aldığında istemci ile haberleşecek "özel" socket oluşturur.

Internet & Internet

i (küçük) → bulunduğu noktadan bağımsız
→ bilgisayar topluluğu

iletişim için IP (Internet Protocol) kullanılır.

I (büyük) → Dünyanın En büyük IP tabanlı ağıdır

(192.168.1.167)
→ Her bilgisayarda tekil IP olur

Protokoller

Echo	7	Test için echo kodu
Daytime	13	Saat bilgisi
FTP-data	20	Dosya gönderme
FTP	21	Komut gönderme
Telnet	23	Uzaktan erişim, komut satırı
SMTP	25	Eposta (Simple mail transfer)
HTTP	80	Hyper Text... (www)
NNTP	119	Usenet. (Network news TP) (Kullanıcı networks)

Notasyon

BNF → [ ] isteğe bağlı

<protocol>:// <host> [:port] [/path] [/file] [#section]

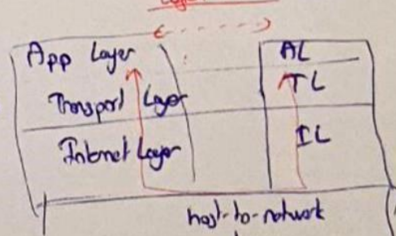
TCP (Transmission Control Protocol)

→ Hand Shaking
data → Sıra no
→ Checksum
→ Full Duplex (eş zamanlı veri gönderimi)

IP (Internet Protocol)

(TCP) → Ip adress

Logical Path



Ethernet [IP [UDP [TFTP [DATA]
↑
veri

Java IO

IO / NIO

I/O için java.io
Dosya I/O için java.nio.file

IO Streams

Byte, Characters, Buffered
Scan & Format, I/O from command line
Data, Object, File Input & Output Stream
tabanlı

❗ işlem bitince stream kapatılmalı
(Sıyı başa alınma ...)

❗ Input & Output Stream doğasından gelirler.

Buffered Stream

→ Flushing ile dolmayan buffer'dan direkt dosyaya yazılır.
→ File read, disk access, network ve maliyetlidir.
→ Str ile dosya okumak yerine parçalı halde alınır maliyetler kesilir.
↳ maliyetli olabilirsen akış var olan "buffer"'a yazdır.

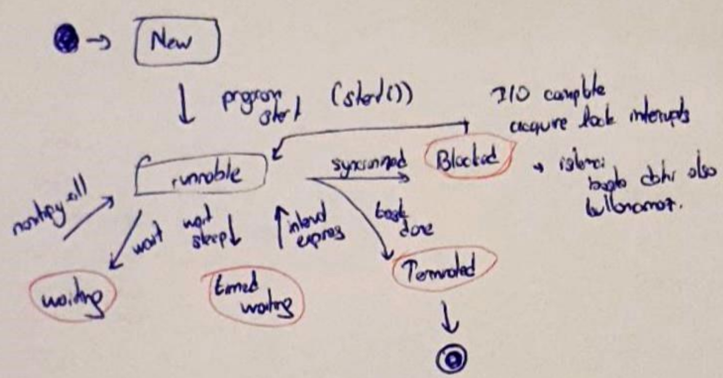
Natural Streams

InputStream Readers
OutputStream Writers
(byte-to-characters

Chr 10

Satır sonunu gösterilsin.
("\r\n") ← Satırı sonlandırma
carriage-return ← line-feed

Threading



→ işlemci başka dahi olsa kullanamaz.

OS için Thread

OS dispatches a thread
Ready → Running
quantum expires

Thread Synchronized

• Ortak kaynakları farklı threadler aynı anda değiştirmemelidir.
↳ Data kontrolü bulanıklaşır.
↳ Thread senkronize edilir.

❗ "Synchronized" ile thread kaynakları "lock" hale alınır, diğer threadler "blocked" durumuna alınır.

fonksiyonu uygularsa tüm block tek tek olarak ele alınır.
(Kaynağı bilerek kullanılanlar)
↳ kilitlenen kaynak ("this")

synchronized (object) {
....
}

↳ Mümkün olduğunca kısa ve "sleep" olmayan kodlar olmalı

Consumer - Producer

• Synchronized olmaksızın (aslında yine var) multithreading yapar.
• Buffer belleği ile veriler saklanır.

Consumer ← (buffer) ← write ← Producer
read

❗ "write" ve "read" threadleri "synchronized" olarak çalışmalı ve "buffer" belleğine "lock" hale almalıdır.

Wait & Notify & Notify All

buffer boş ise yazı dolu ise "wait"

Consumer ← buffer ← Producer

Boş ise "wait"
dolu ise "read"

❗ « out of sync » durumunda threadler "wait" durumunda tutulabilir.
↳ bounded buffer

Bounded Buffer

❗ Producer & Consumer hızları aynı farklı ise ~~gecikme~~ "yeterli" olacaktır.

"ArrayBlockingQueue" "constructor" ında sınırlaması ile tanımlanır.

java.util.concurrent

ArrayBlockingQueue

• Thread-safe > BlockingQueue
• Ortak buffer'ın sistemi kontrol edecek olan yapısı
"put" "take"

Java Multi Threading

ReentrantLock

"Lock" interface'ini implemente etmiştir.
"fairness policy" değeri "constructor" ile verilir.
"True" ise en uzun bekleyen thread "lock"u alır.

Lock Interface Avantajı

→ "condition"

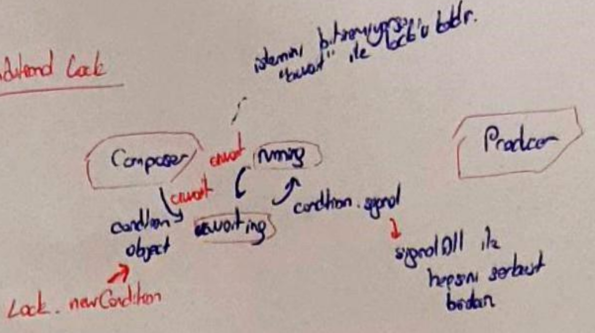
Thread'i interrupt etme
"timeout" "lock"

Kodların, Aynı blok içerisinde bulunmasına gerek yok

Java ile Ağ Programlama

- "java.net" paketi kullanılır
- "InetAdress" objesi ile ip adresiyle host adı alınır.
- "getByName" ile DNS bilgisinden IP adresini "InetAdress" objesi olarak verir.
- "getLocalHost" ile local IP adresi alınır.

Conditional Lock



GUI & Multithread

- "Swing" kütüphanesi kullanılabilir
- GUI istekleri "event dispatch thread" isimli tek thread ile yapılır. (thread confinement)
- Arayüz istekleri "event queue" ile sırayla yapılır.
  - Bileşenler (GUI) "non-thread-safe"
  - "syncronized" yok.

Swing

→ uzun olan işleri arkaplanda yapar

"doInBackground"
"done" → doInBack. bitince "event dispatch thread" ile tetiklenir.
"execute" → "Swing Worker" objesi ile worker thread üzerinde çalıştırır
"get"
"publish"
"process"
"setProcess"

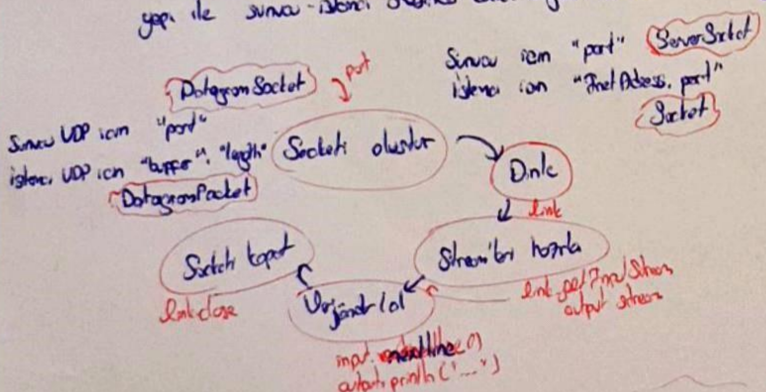
doInBack. üzerindeki ara düzeyli mesajları "process"e gönderir

"publish" ten alınanları "event dispatch thread" üzerinde çalıştırır

Özellik değişimini dinleyen objeleri (progress bar) gibi) tetikler ve bildirim verir.

TCP Socket

- "connection-oriented" yani bağlantılı yönelimli yapı ile sunucu-istemci arasında kalıcı bağlantı kurulur.



Datagram Socket (UDP)

- "connectionless" bağlantısız, sadece veri davranışına odaklı yapı
- Hızlı ama güvenilmez yapı (TCP'ye göre)
- İletilen paketler "Datagram" olarak bağımsız iletilirler. (bağlantısız)
- "DatagramSocket" kullanılır.

UDP Sunucu (Loop)

udp için:
dotagramSocket.receive(inDatagramPacket)

! "IP" ve "port" bilgisi "DatagramPacket" üzerinden alınır.

- Bufferdan veri alınması
  String message = new String(inPacket.getData(), 0, inPacket.getLength());
- Cevap "datagram"
  DatagramPacket outPacket = new DatagramPacket(response.getBytes(), response.length(), clientAdress, clientPort);
- Gönderme
  datagramSocket.send(outPacket)
- Socketi kapatma
  datagramSocket.close();

Client UDP

- Obje oluşturma
  DatagramSocket datagramSocket = new D.Sock..
- Çıkış cevabı
  DatagramPacket outPacket = new DatagramPacket(message.getBytes(), message.length(), host, port);
- Gönderme
  datagramSocket.send(outPacket);
- Gelen datagramlar için paket objesi oluşturma
  DatagramPacket inPacket = new DatagramPacket(buffer, buffer.length());
- Okuma (Datagram)
  datagramSocket.receive(inPacket)
- Buffer okuma
  String response = new String(inPacket.getData(), 0, inPacket.getLength());
- Socket'i kapatma
  datagramSocket.close()

Loop UDP Client