Mrezno računarstvo

4.-6. glava

Looking Up Internet Addresses

pojmovi

- *node* – uređaj povezan na Internet
- *host* – node koji je računar
- svaki host se identifikuje bar jednim jedinstvenim brojem koji se naziva *Internet adresa ili IP adresa*
- IPv4 adrese – 4 bajta
- IPv6 adrese – 16 bajtova
- (4 i 6 su verzije Internet Protokola, ne brojevi bajtova u adresi)
- adrese su uređeni nizovi bajtova, nisu brojevi

IPv4 adresa

- 4 neoznačena bajta, svaki sa vrednošću od 0 do 255. Bajtovi se razdvajaju tačkama
- 152.2.21.2 (dotted quad format)

IPv6 adresa

- 8 blokova od po 4 heksadekadne cifre
- blokovi se razdvajaju dvotačkom
- 2001:0250:02FF:0210:0250:8BFF:FEDE:67C8
- vodeće nule ne moraju se pisati
- 2001:250:2FF:210:250:8BFF:FEDE:67C8
- uzastopna dvotačka, najviše jedna u adresi, ukazuje na pojavu višestrukih 0-blokova
- FEDC:0000:0000:0000:00DC:0000:7076:0010
- FEDC::DC:0:7076:10

Mešovite IPv6 i IPv4 mreže

- poslednja 4 bajta IPv6 adrese se ponekad pišu kao IPv4 dotted quad adresa:
- FEDC:BA98:7654:3210:FEDC:BA98:7654:3210
- FEDC:BA98:7654:3210:FEDC:BA98:118.84.50.16
- IPv6 je podržan od verzije Java 1.4

DNS

- *Domain Name System (DNS)* pridružuje IP adresama hostname-ove koje čovek može da upamti
- Većina host-ova ima bar jedan hostname
- Izuzetak su računari koji nemaju stalnu IP adresu. Pošto nemaju stalnu IP adresu, takvi računari se ne mogu koristiti kao serveri, pa nema potrebe da imaju ime jer im se niko neće obraćati
- Praviti razliku: IP adresa je uvek numerička, a hostname je human-readable

DNS

- Neke mašine imaju veći broj imena.
- Npr. www.ibiblio.org i helios.metalab.unc.edu,

(Da li to vazi za www.math.rs i poincare.matf.bg.ac.rs?)

- www.ibiblio.org se odnosi pre na web site nego na mašinu
- u prošlosti, kada se web site premeštao sa mašine na mašinu, ime je repridruživano novoj mašini tako da uvek pokazuje na tekući server sajta. Na ovaj način, URL-ovi širom sveta ne moraju se ažurirati samo zato što je neki sajt pomeren na novi host.

DNS

- povremeno, 1 ime se mapira na veći broj IP adresa. Tada je DNS server odgovoran da slučajno bira mašine da odgovore na svaki zahtev.
- Ovo se često koristi za dosta opterećene web sajtove, gde se opterećenje deli na veći broj sistema
- Svaki računar povezan na Internet treba da ima pristup mašini koja se zove *domain name server*, koja je u opštem slučaju pod Unix-om i izvršava poseban DNS software koji zna mapiranja između različitih hostname-ova i IP adresa

DNS

- Većina domain name servera zna samo adrese hostova u svojoj lokalnoj mreži i adrese nekoliko drugih domain name servera. Kada klijent traži adresu mašine izvan lokalnog domena, local domain server traži podatak od udaljenog domain servera i prosleđuje odgovor klijentu
- veći deo vremena koristimo hostname-ove i puštamo da DNS rukuje prevodom IP adresa.
- Za pokretanje primera iz knjige, neophodno je da računar bude povezan na Internet (i time bude u vezi sa bar jednim DNS-om)

klasa InetAddress

- `java.net.InetAddress`
- reprezentacija visokog nivoa IP adrese (IPv4, IPv6)
- Koristi je većina drugih mrežnih klasa, uključujući `Socket`, `ServerSocket`, `URL`, `DatagramSocket`, `DatagramPacket`...
- uključuje i hostname i IP adresu
- `public class InetAddress extends Object implements Serializable`
- (iz ove klase ne treba izvoditi svoje klase, a to nije ni moguće jer su svi konstruktori sa paketnim pravom pristupa)

Kreiranje novih InetAddress objekata

- ne postoje public konstruktori klase
- postoje 3 statička metoda koja vraćaju pogodno inicijalizovane InetAddress objekte
- `public static InetAddress getByName(String hostName) throws UnknownHostException`
- `public static InetAddress[] getAllByName(String hostName) throws UnknownHostException`
- `public static InetAddress getLocalHost() throws UnknownHostException`

Kreiranje novih InetAddress objekata

- Sva 3 ova metoda mogu napraviti konekciju sa lokalnim DNS kako bi upotpunili informacije o InetAddress objektu, ako je neophodno
- Ovi metodi mogu izbaciti ”security” izuzetke ako je zabranjena konekcija sa DNS-om
- Ključna stvar za zapamtiti jeste da ovi metodi prosto ne koriste svoje argumente kako bi postavili odgovarajuća polja objekta
- Oni zapravo *prave mrežne konekcije kako bi dobili neophodne informacije*
- Ostali metodi klase, poput getAddress() i getHostName() uglavnom rade sa informacijama koje su obezbedili ovi metodi. Ti metodi ne prave mrežne konekcije, a u retkim situacijama kada to rade, ne izbacuju izuzetke
- *Samo ova 3 metoda idu izvan Jave i lokalnog sistema kako bi obavili svoj posao*

- DNS pretrage mogu biti relativno skupe (reda veličine nekoliko sekundi po zahtevu koji ide preko nekoliko međuservera, ili zahtevu koji pokušava da razreši host koji nije dostupan)
- Zato klasa InetAddress *kešira rezultate pretraga*
- Ako ima adresu datog host-a, neće je tražiti ponovo, čak i ako se kreira novi InetAddress objekat za isti host. Sve dok se IP adrese ne menjaju za vreme izvršavanja, ovo ne predstavlja problem

- greške tipa: host not found su neznatno problematičnije. Nije neuobičajeno da inicijalni pokušaj razrešenja host-a ne uspe, ali naredni koji neposredno sledi uspe. Ono što se obično dešava u ovoj situaciji je da prvi pokušaj istekne dok su informacije još u putu od DNS servera. Zatim informacija stiže do lokalnog servera i odmah je dostupna za sledeći zahtev. Zato, Java *neuspešne zahteve kešira na 10 sekundi*.
- postoje system properties kojima se može podesiti broj sekundi koliko se uspešne i neuspešne pretrage čuvaju u kešu

- Osim lokalnog keširanja unutar klase InetAddress, local host, local domain name server i drugi DNS serveri gdegod na Internetu takođe mogu keširati rezultate različitih upita. Java nema kontrolu nad tim.
- Posledica je da može proći nekoliko sati dok informacija o promeni IP adrese ne prođe kroz Internet. U međuvremenu naši programi se mogu suočiti sa različitim izuzecima, uključujući `UnknownHostException`, `NoRouteToHostException` i `ConnectException` u zavisnosti od načinjene promene

- 2 metoda koja *ne proveravaju* svoje adrese kod lokalnog DNS servera
- prvi kreira InetAddress objekat sa zadatom IP adresom i bez hostname
- drugi kreira InetAddress objekat sa IP adresom i hostname-om
- `public static InetAddress getByAddress(byte[] address) throws UnknownHostException`
- `public static InetAddress getByAddress(String hostName, byte[] address) throws UnknownHostException`
- Za razliku od prethodna 3, ova 2 metoda ne garantuju da takav host uopšte postoji ili da je hostname korektno mapiran na IP adresu. *Izbacuju izuzetak jedino ako je niz bajtova nekorektne veličine* (nije 4 niti 16 bajtova) prosleđen kao argument address

public static InetAddress getByName(String hostName) throws UnknownHostException

- najčešće korišćen
- statički metod koji uzima hostname koji tražimo kao argument
- traži IP adresu tog hosta koristeći DNS.
- `import java.net.*; (biće u svim programima)`
- ```
  try{
      InetAddress address =
          InetAddress.getByName("www.oreally.com");
      System.out.println(address);
   }catch(UnknownHostException ex){
      System.out.println("Could not find www.oreilly.com");
   }
  ```
- primer1
- u retkim prilikama želimo da se povežemo sa mašinom koja nema hostname. Moguće je proslediti String koji sadrži dotted quad ili heksadekadni oblik IP adrese kao argument getByName() metoda
  ```

InetAddress ia = InetAddress.getByName("208.201.239.100");

- kada se getByName() poziva sa stringom u kome je IP adresa, on kreira InetAddress objekat za zadatu IP adresu bez provere DNS-a. To znači da je moguće kreirati InetAddress objekte za host-ove koji ne postoje i nemoguće je konektovati se sa njima.
- DNS pretraga za odgovarajući hostname se vrši samo kada se hostname zahteva pozivom metoda `getHostName()`
- `toString()` ne vrši potragu za hostname, pa host neće biti uključen u String-reprezentaciju, osim ako je već poznat, bilo zato što je prosleđen kao argument metoda prilikom kreiranja InetAddress objekta ili zato što je pozivan metod getHostName()

IP adresa ili hostname?

- hostname-ovi su dosta stabilniji od IP adresa
- Neki servisi su živeli na istom hostname-u nekoliko godina a menjali su IP adrese nekoliko puta.
- Ako imate izbor između korišćenja hostname-a i IP-adrese, uvek izaberite hostname i koristite IP adresu samo kada hostname nije dostupan.

public static InetAddress[] getAllByName(String hostName) throws UnknownHostException

- neki računari imaju više od jedne Internet adrese
- za zadati hostname, ovaj metod vraća niz koji sadrži sve adrese koje odgovaraju tom imenu
- primer2 vraća kompletnu listu IP adresa za "www.microsoft.com" (sada ima samo jednu -> probati za "www.google.com", on ih ima više)
- host-ovi sa više od jedne adrese su pre izuzetak nego pravilo. Većina su veliki web serveri. Čak i tada retko je potrebno znati više od jedne njihove adrese

public static InetAddress getByAddress(byte[] address) throws UnknownHostException

public static InetAddress getByAddress(String hostName, byte[] address) throws UnknownHostException

- kreiranje InetAddress objekta sa tačno zadatim argumentima.
- Ne vrši se domain name pretraga
- Ako je dužina niza različita od 4 ili 16 izbacuje se izuzetak
- korisno kada domain name server nije dostupan ili ima netačnu informaciju

public static InetAddress getLocalHost() throws UnknownHostException

- vraća InetAddress objekat za mašinu na kojoj se izvršava
- izbacuje `UnknownHostException` kada ne može da nađe adresu lokalne mašine (mada ovo ne bi trebalo da se desi)
- upotreba je pravolinijska

```
InetAddress me =
  InetAddress.getLocalHost();
```

- primer3
- ako nismo konektovani na Internet i sistem nema fiksnu IP adresu ili domain name, verovatno ćemo videti localhost kao domain name i 127.0.0.1 kao IP adresu

Security Issues

- Kreiranje novog InetAddress objekta od zadatog hostname smatra se potencijalno nebezbednom operacijom jer zahteva DNS pretragu
- apletu pod kontrolom podrazumevanog security manager-a jedino je dopušteno da dobije IP adresu hosta sa kog dolazi (svog codebase) i moguće local host-a.
- Nije dopušteno da kreira InetAddress objekat od bilo kog drugog hostname-a. Može od string oblika IP adrese jer se tada ne vrši DNS pretraga za takvom adresom

- Nepoverljivom kodu nije dopušteno da vrši proizvoljne DNS pretrage za drugim host-ovima zbog zabrane pravljenja konekcija sa host-ovima osim codebase. Proizvoljne DNS pretrage otvorile bi tajne kanale kojim bi program mogao komunicirati sa ostalim host-ovima. (curenje informacija)
- Nepoverljivom kodu je dopušteno da zove `InetAddress.getLocalHost()`. Međutim, ovaj metod vraća hostname localhost i IP 127.0.0.1 Ovaj specijalan hostname i IP zovu se loopback adresa. Bez obzira na to koja mašina se koristi, ovaj hostname i IP adresa uvek se odnose na tekuću mašinu. Nije neophodno DNS razrešenje.
- Razlog zabrane apletu da otkrije stvarni hostname i adresu je to što je računar na kome se aplet izvršava namerno skriven iza firewall-a. U tom slučaju, aplet ne sme biti kanal za informacije koje nema web server.

- Kao i sve sigurnosne provere, zabrane DNS razrešenja mogu biti oslabljene za kod kome verujemo.
- metod klase `SecurityManager` kojim se proverava da li host može biti razrešen:

```
public void checkConnect(String hostname,
                         int port)
```

- kada je port -1, metod proverava da li se može pozvati DNS da razreši zadati host
- ako je port veći od -1, metod proverava da li je dopuštena konekcija sa imenovanim host-om na zadatom portu
- host može biti bilo hostname, bilo dotted quad IP adresa, bilo heksadekadna IPv6 adresa
- Ako je razrešenje/konekcija dopuštena, metod radi return, a inače izbacuje `SecurityException`
- apletu se može dodeliti pravo da razreši host-a korišćenjem Policy Tool
- primer 3a (InetAplet) + 3a_dodatna_uputstva.pdf

getter metodi klase InetAddress

- `public String getHostName()`
- `public byte[] getAddress()`
- `public String getHostAddress()`
- ne postoje odgovarajući set*() metodi
- nema načina da se izvan paketa java.net promene polja InetAddress objekta
- Java može garantovati da hostname i IP adresa odgovaraju jedno drugom
- InetAddress objekat je nepromenljiv i thread-safe

public String getHostName()

- vraća String koji je ime host-a sa IP adresom predstavljenom InetAddress objektom
- ako mašina nema hostname ili security manager sprečava određivanje imena, metod vraća dotted quad format numeričke IP adrese
- InetAddress machine = InetAddress.getLocalHost();

  String localHost = machine.getHostName();
- Nekad se vidi samo delimično kvalifikovano ime. To zavisi od načina na koji se DNS ponaša kada razrešava local hostname-ove
- primer4

public String getHostAddress()

- vraća String koji sadrži dotted quad format IP adrese
- primer5

public byte[] getAddress()

- (retko) IP mašine kao niz bajtova u mrežnom poretku. Bajt najveće težine(krajnje levi u dotted quad form) je prvi bajt u nizu, tj. element sa indeksom 0
- Zbog IPv6 adresa, ne pretpostavljati ništa o dužini niza. Ako želimo da znamo tu dužinu, možemo iskoristiti polje length:
- InetAddress me = InetAddress.getLocalHost();

  byte[] address = me.getAddress();
- Vraćeni bajtovi su neoznačeni, što može uzrokovati problem. U Javi ne postoji tip neoznačeni bajt kao u C-u. Bajtovi sa vrednostima većim od 127 tretiraju se kao negativni brojevi. Ako želimo bilo šta da radimo sa bajtovima koje nam je vratio getAddress() metod, moramo ih kastovati u int-ove i prilagoditi ih. Jedan način je:
- int unsignedByte = signedByte < 0 ? signedByte + 256 : signedByte;
- Ovde signedByte može biti pozitivan ili negativan
- Jedan od razloga da se dobije neobrađeni niz bajtova IP adrese je određivanje tipa adrese. Testiranjem broja bajtova koje je vratio getAddress() određuje se da li se radi sa IPv4 ili IPv6 adresom
- Primer6

Tipovi adresa

- Neke adrese imaju specijalno značenje
- 127.0.0.1 je lokalna loopback adresa
- IPv4 adrese iz opsega 224.0.0.0 do 239.255.255.255 su multicast adrese koje šalju većem broju host-ova odjednom
- 10 metoda za testiranje da li InetAddress objekat zadovoljava neki od kriterijuma
- isAnyLocalAddress()
  isLoopbackAddress()
  isLinkLocalAddress()
  isSiteLocalAddress()isMultiCastAddress()
  isMCGlobal() isMCNodeLocal()
  isMCLinkLocal() isMCSiteLocal()
  isMCOrgLocal()

- public boolean isAnyLocalAddress()
- true ako je adresa wildcard adresa, a wildcard adresa odgovara proizvoljnoj adresi na lokalnom sistemu. Ovo je važno kada sistem ima nekoliko mrežnih interfejsa (nekoliko Ethernet karti ili Ethernet kartu i wireless konekciju). Obično je ovo bitno samo na serverima i gateway-ima. U IPv4 wildcard adresa je 0.0.0.0
- U IPv6 je 0:0:0:0:0:0:0:0 (ili ::)

- public boolean isLoopbackAddress()
- loopback adresa se konektuje na isti računar *direktno u IP sloju ne koristeći nikakav fizički hardware*
- konektovanje na loopback adresu omogućuje testiranje i pomaže u detektovanju problema
- konektovanje na loopback adresu nije isto što i konektovanje na stvarnu IP adresu sa istog sistema. U IPv4 ova adresa je 127.0.0.1, a u IPv6 je 0:0:0:0:0:0:0:1 (::1)

Testing Reachability

- 2 metoda koja omogućuju aplikacijama da testiraju da li je određeni node dostupan tekucem host-u, tj. da li je moguće napraviti mrežnu konekciju
- Konekcije mogu biti blokirane iz više razloga, uključujući firewall-ove, proxy servere, pokvarene rutere i prekinute kablove ili jednostavno to što je udaljeni host isključen kada probamo da se konektujemo.
- public boolean isReachable(int timeout) throws IOException
- public boolean isReachable(NetworkInterface interface, int ttl, int timeout) throws IOException
- ovi metodi pokušavaju da se konektuju na echo port udaljenog host-a kako bi saznali da li je dostupan
- ako host odgovori za timeout milisekundi, metod vraća true
- izuzetak se izbacuje ako postoji mrežna greška
- ”time-to-live“ – maksimalan broj mrežnih skokova koja konekcija pokuša pre odbacivanja
- local network interfejs od kog je napravljena konekcija
- u praksi, ovi metodi nisu vrlo pouzdani kroz globalni Internet. U lokalnom intranetu mogu se koristiti

metodi klase Object

- predefiniše sledeća 3 metoda
- `public boolean equals(Object o)`
- `public int hashCode()`
- `public String toString()`

- equals()
- objekat je jednak datom InetAddress objektu samo ako je i sam instanca klase InetAddress i ima istu IP adresu. Ne mora imati isti hostname
- primer7

- public int hashCode()
- vraća int potreban kada se InetAddress objekti koriste kao ključevi u heš tabelama
- vrednost se računa samo na osnovu IP adrese, ne uzima u obzir hostname
- Ako dva InetAddress objekta imaju istu adresu, imaju isti hash code, čak i ako im se hostname-ovi razlikuju

- public String toString()
- hostname/dotted quad address
- nemaju svi InetAddress objekti hostname-ove. Ako neki objekat nema hostname, on se postavlja na prazan string.

Klase Inet4Address i Inet6Address

- final i izvedene iz InetAddress klase
- nisu potrebne
- u application sloju, gde se izvršavaju Java programi, ne moramo znati da li je adresa IPv4 ili IPv6, a čak i da moramo, brže je proveriti veličinu niza bajtova koje vrati getAddress() nego koristiti instanceof za testiranje koja potklasa je u pitanju

klasa NetworkInterface

- *predstavlja lokalnu IP adresu*
- to može biti fizički interfejs kao što je dodatna Ethernet karta (uobičajeno na firewall-ovima i ruterima) ili može biti virtualni interfejs vezan za fizički hardware kao što su druge IP adrese mašine
- ova klasa obezbeđuje metode za enumeraciju svih lokalnih adresa i kreiranje InetAddress objekata od njih
- ovi objekti se potom mogu koristiti za kreiranje soketa, server soketa itd.

metodi klase NetworkInterface

- pošto predstavljaju fizički hardware i virtualne adrese, ne mogu se proizvoljno konstruisati
- postoje statički metodi koji vraćaju NetworkInterface objekat pridružen određenom mrežnom interfejsu
- možemo dobiti NetworkInterface objekat pomoću IP adrese, imena ili enumeracije

- public static NetworkInterface getByName(String name) throws SocketException
- vraća NetworkInterface objekat koji predstavlja mrežni interfejs sa zadatim imenom. Ako nema interfejsa sa tim imenom, vraća `null`
- ako mrežni stek otkrije problem, izbacuje se izuzetak, ali nije puno verovatno da se to desi
- Format imena je platformski zavisan. Na tipičnom Unix sistemu, Ethernet interfejs imena su oblika eth0, eth1 itd. Lokalna loopback adresa se obično zove ”lo”. Na Windows-u, imena su stringovi poput ”CE31” ili ”ELX100” izvedeni iz imena proizvođača i modela hardware-a za određeni mrežni interfejs.

- public static NetworkInterface getByInetAddress(InetAddress address) throws SocketException
- vraća NetworkInterface objekat koji predstavlja mrežni interfejs vezan za određenu IP adresu. Ako nema mrežnog interfejsa vezanog za tu IP adresu na lokalnom hostu, vraća `null`. Ako bilo šta krene naopako, izbacuje se izuzetak.
- primer8

- public static Enumeration getNetworkInterfaces() throws SocketException
- vraća `java.util.Enumeration` listu svih mrežnih interfejsa na lokalnom host-u.
- primer9
- ignorisati broj adresa. to je broj bez značenja, ne stvarni broj IP adresa vezanih za svaki interfejs (u Java 7 taj broj se i ne prikazuje)

getter metodi

- kada imamo NetworkInterface objekat, možemo dobiti njegovu IP adresu i ime
- to je otprilike sve što sa njim možemo da radimo
- public Enumeration getInetAddresses()
- jedan mrežni interfejs može biti vezan za više od jedne IP adrese
- ovaj metod vraća enumeraciju koja sadrži InetAddress objekat za svaku IP adresu za koju je interfejs vezan
- public String getName() vraća ime određenog NetworkInterface objekta, poput eth0 ili lo
- public String getDisplayName() vraća čitljivije ime za određeni NetworkInterface

metodi klase Object

- `equals()`, `hashCode()`, `toString()`
- NetworkInterface objekti su jednaki ako predstavljaju isti fizički mrežni interfejs (oba ukazuju na isti Ethernet port, modem ili wireless kartu) i imaju istu IP adresu. Inače su različiti
- NetworkInterface ne implementiraju `Cloneable`, `Serializable` ili `Comparable` interfejse

Neki korisni programi

- dva primera: jedan koji interaktivno postavlja upite domain name serveru, a drugi koji može poboljšati performanse web servera procesiranjem log fajlova offline

primer10: HostLookup

- konvertovanje hostname-ova u IP adrese i IP adresa u hostname-ove
- ima 2 moda: interaktivni i command-line
- ako se unese hostname u komandnu liniju, štampa se IP adresa tog host-a.
- ako se unese IP adresa u komandnu liniju, štampa se hostname
- ako u komandnoj liniji nema ni hostname-a ni IP adrese, ulazi se u interaktivni mod gde se čitaju hostname-ovi i IP adrese sa standardnog ulaza a ispisuju odgovarajuće IP adrese i hostname-ovi dok se ne unese ”exit”.

primer 10, objašnjenja

- 3 metoda: main(), lookup() i isHostName()
- main() metod određuje ima li argumenata komandne linije
- ako ih ima, main() poziva lookup() za svaki od njih
- inače, olančava BufferedReader na InputStreamReader olančan na System.in i čita ulaz metodom readLine() (ovde program čita iz konzole, ne iz mrežne konekcije)
- ako je linija ”exit”, program se završava, a inače se linija prosleđuje metodu lookup()

- lookup() metod koristi InetAdress.getByName() da odredi traženi host, bez obzira na format ulaza. getByName() prihvata i ime i dotted quad address.
- lookup() poziva isHostName() da odredi da li je string hostname, dotted quad IPv4 adresa ili IPv6 adresa.
- isHostName() prvo traži dvotačke, koje ima svaka IPv6 adresa, a hostname nema. Ako nađe dvotačku, vraća false. Provera IPv4 adresa je komplikovanija jer ne sadrži nijedan karakter koji se ne može pojaviti u hostname. Metod gleda svaki karakter stringa pa ako su svi cifre i tačke, pretpostavlja da je string numerička IP adresa i vraća false. Inače, pretpostavlja da je string hostname i vraća true.
- Šta ako nije ni jedno ni drugo? getByName() neće moći da ga reši pa će izbaciti izuzetak.
- Ako korisnik unese hostname, lookup() vraća odgovarajuću dotted quad ili heksadekadnu adresu koristeći getHostAddress().
- Ako korisnik unese IP adresu, koristi se metod getHostName() za nalaženje odgovarajućeg hostname-a.

Primer11: Processing Web Server Log Files

- web serveri prate koji hostovi pristupaju sajtu
- podrazumevano, registruju se IP adrese koje se konektuju na server
- međutim, često je moguće dobiti više informacija iz imena nego iz odgovarajuće IP adrese.
- Većina servera ima opciju da čuva hostname-ove umesto IP adresa, ali to može narušiti performanse jer server mora da traži DNS zahtev za svaku adresu.
- Mnogo je efikasnije logovati IP adrese i konvertovati ih u hostname-ove kasnije, kada server nije zauzet, ili čak na nekoj potpuno drugoj mašini
- Primer čita log fajl web servera i štampa linije u kojima je IP adresa konvertovana u hostname

The Common Log File Format

- Većina servera ima standardizovani format log fajla
- Tipična linija tog fajla izgleda ovako:
- 205.160.186.76 unknown – [17/Jun/2003:22:53:58 -0500] ”GET 7bgs/greenbg.gif HTTP 1. 0” 200 50
- web browser na IP adresi 205.160.186.76 zahteva fajl /bgs/greenbg.gif sa ovog web servera u 10:53 p.m. (i 58 sekundi) 17. juna 2003.
- fajl je pronađen (response code 200) i 50 bajtova podataka je uspešno preneto browser-u.

- Prvo polje je IP adresa ili, ako je uključeno razrešenje DNS-a, hostname sa kog je napravljena konekcija. Za ovim sledi blanko
- Za naše svrhe, parsiranje log fajla je jednostavno: sve do prvog blanka je IP adresa, a sve posle toga ne treba menjati

Primer11, objašnjenja

- Ime fajla koji se procesuira prosleđuje se programu kao prvi argument komandne linije
- otvara se `FileInputStream` za taj fajl i na njega olančava `InputStreamReader`. On se baferiše olančavanjem na njega instance klase `BufferedReader`.
- Fajl se obrađuje linija po linija u while-petlji
- Svaki prolaz kroz petlju smešta jednu liniju u `String` promenljivu entry.
- entry se zatim deli u dva podstringa:
- ip – koji sadrži sve pre prvog blanka
- theRest – sve nakon prvog blanka
- pozicija prvog blanka određuje se metodom `indexOf()`
- ip se konvertuje u InetAddress koristeći `getByName()`
- `getHostName()` zatim traži hostname
- konačno, hostname, blanko i ostatak linije se štampaju na std. izlaz

- Program je efikasniji nego što bi se dalo očekivati. Većina web servera generiše višestruke unose u log fajlu za jednu stranicu, pošto postoji po unos, ne samo za stranicu, već i za svaku sliku na stranici
- mnogi posetioci traže veći broj strana sa sajta
- DNS pretrage su skupe i nema smisla tražiti svaki sajt svaki put kada se pojavi u log fajlu
- klasa InetAddress vrši keširanje traženih adresa. Ako se ista adresa traži ponovo, može se dohvatiti iz keša mnogo brže nego pomoću DNS-a

Primer12: Ubrzanje, thread pool

- Program može biti brži.
- obrada traje više od sekunde po unosu
- tačna brzina zavisi od brzine mrežne konekcije, brzine lokalnog i udaljenog DNS servera i opterećenja kada se izvršava program
- ogromne količine vremena program troši čekajući da se vrate zahtevi DNS-a
- ovo je upravo problem koji se rešava multithreadingom
- jedna glavna nit može čitati log fajl i prosleđivati pojedinačne unose drugim nitima na procesiranje

- thread pool je ovde apsolutno neophodan
- za nekoliko dana čak i manje opterećeni web serveri mogu lako generisati log fajl sa stotinama hiljada linija
- pokušaj obrade fajla generisanjem nove niti za svaki unos nije rešenje, posebno pošto glavna nit može čitati unose mnogo brže nego što pojedinačne niti mogu razrešavati domain name-ove.
- Dakle, reusing niti je ključna stvar.
- Broj niti se čuva u parametru numberOfThreads tako da se može prilagoditi VM i mrežnom steku
- Pokretanje prevelikog broja simultanih DNS zahteva takođe može izazvati probleme

PooledWeblog klasa

- Program je podeljen u dve klase
- prva, PooledWeblog sadrži metod main() i processLogFile()
- takođe čuva resurse koji treba da budu deljeni od strane niti
- to su: pool, implementiran kao sinhronizovana `LinkedList` i izlazni log out implementiran kao `BufferedWriter`
- Pojedinačne niti imaju direktan pristup pool-u, ali moraju da prođu kroz metod log() kako bi pisale na izlaz.

- Ključni je metod processLogFile() koji, kao i ranije, čita iz log fajla.
- međutim, svaki unos se smešta u pool entries umesto da se neposredno obrađuje.
- pošto je verovatno da se ovaj metod izvršava mnogo brže od niti koje pristupaju DNS-u, on radi yield nakon svakog unosa.
- Dalje, on ide da spava ako ima više unosa u pool-u nego niti koje su raspoložive za njihovo procesiranje. Količina vremena koje provede spavajući zavisi od broja niti.
- Kada se pročita poslednji unos, fleg finished se postavlja na true kako bi kazao nitima da mogu da umru nakon što završe svoj posao.

LookupThread klasa

- rukuje konvertovanjem IP adresa u hostname-ove u unosima log fajla
- konstruktor svakoj niti obezbeđuje referencu na pool entries iz kog uzima svoj posao i referencu na PooledWeblog objekat za koji radi
- druga referenca omogućava callback-ove tako da nit može upisati konvertovane unose u log fajl i proveriti da li je obrađen i poslednji unos. Ona vrši tu proveru pozivom metoda isFinished() kada je pool entries prazan (tj. veličina mu je 0)
- ni prazan pool ni isFinished() koji je vratio true nisu sami po sebi dovoljni. isFinished() vraća true kada se poslednji unos stavi u pool, a može proteći bar mala količina vremena dok se on iz njega ukloni. entries može biti prazan iako ima još mnogo unosa koje treba pročitati ako pojedinačne niti odrade posao brže nego što glavna nit čita log fajl
- Na ovaj način, korišćenjem niti, postižu se ogromne uštede u vremenu – 10 do 50 puta brže od sekvencijalne verzije programa
- najveća mana: reorganizacija log fajla (nije nužno isti redosled unosa)