Računarske mreže, Ispit - JAN1

18.01.2021.

Pročitati sve zadatke **pažljivo** pre rada - sve što nije navedeno ne mora da se implementira!
Na `Desktop`-u se nalazi zip arhiva. Unutar arhive se nalazi direktorijum u formatu `rm_rok_Ime_Prezime_mXGGXXX` u kome se nalazi validan IntelliJ projekat. Izvući direktorijum iz arhive na Desktop i ubaciti svoje podatke u ime. Otvoriti IntelliJ IDEA, izabrati opciju `Open project` (ne `Import project`!) i otvoriti pomenuti direktorijum. Sve kodove ostaviti unutar već kreiranih Java fajlova. **Kodovi koji se ne prevode se neće pregledati. Nepoštovanje formata ulaza/izlaza nosi kaznu od -10% poena na zadatku!** Vreme za rad: **3h**.

1. **Da Vinčijev URL (20p)**
   Napisati program koji na standardni izlaz ispisuje informacije o skrivenim URL-ovima. U direktorijumu tests na Desktop-u nalaze se datoteke koje sadrze informacije o delima Leonarda da Vinčija — svaki red je jedna informacija. Svaka datoteka osim informacija sadrži i redove koji predstavljaju validane URL-ove. Ali kako je Leonardo poznat po svojim šiframa datoteke se sastoje od linija takvih da je svaka linija ponaosob napisana "unazad"(zdesna nalevo), pa je linije prvo potrebno "dešifrovati".

   - Obići direktorijum `tests` i za svaku datoteku pokrenuti zasebnu nit koja će obraditi tu datoteku (3p)
   - Za liniju koja "dešifrovana"predstavlja validan URL kreirati novi URL objekat koristeći URL klasu, ostale linije preskočiti (2p)
   - Za svaku datoteku ispisati redom sledeće informacije
     - naziv datoteke (1p)
     - za svaki **validni** URL ispisati broj linije u kojoj je pronađen i protokol koji se koristi (format ispisa: `broj_linije : protokol`) (5p)
     - ako je protokol `https` ispisati i port (2p)
     - koliko linija datoteke predstavlja validan URL (1p)

     (Pogledati primer ispisa ispod)
   - Postarati se da se ispisi svake niti na standardni izlaz ne prepliću (4p)
   - Postarati se da program ispravno obrađuje specijalne slučajeve i ispravno zatvara sve korišćene resurse (2p)

```
Spisak datoteka u direktorijumu tests:
    \home\ispit\Desktop\tests\LadyWithAnErmine.txt
    \home\ispit\Desktop\tests\LastSupper.txt
    \home\ispit\Desktop\tests\MonaLisa.txt
    \home\ispit\Desktop\tests\VitruvianMan.txt

Primer ispisa za LastSupper.txt:
    LastSupper
    5 : file
    8 : https : -1
    2
```

Napomena: Ohrabrujemo studente da koriste `netcat` kako bi testirali delimične implementacije i otkrili greške pre vremena. Takodje, ukoliko se npr. preskoči implementacija servera, može se mock-ovati server putem `netcat`-a.

Okrenite stranu!

2. **Odrad.io (TCP) (25p)**
Zajednički život i održavanje domaćinstva sa sobom donosi razne frustracije, pogotovo u periodu ispitnog roka. Srećom, u frižideru Vas čeka parče omiljene torte koje planirate da pojedete na pauzi spremanja ispita iz Računarskih mreža. Međutim, shvatate da nema čistih kašika, jer Vaš cimer opet nije oprao sudove! Inspirisani novostečenim znanjem iz RM, odlučujete da napravite aplikaciju za praćenje kućnih poslova.

- Implementirati klijentsku TCP aplikaciju koristeći *Java Socket API*. Klijent se povezuje na server na portu 12345 i predstavlja se slanjem svog imena (jedna reč) koje se učitava sa standardnog ulaza. Nakon toga, klijent može poslati neku od tri komande: (8p)
  - `odradi`
  - `dodaj <rec_koja_opisuje_zadatak>`
  - `izadji`

  Komanda se učitava sa standardnog ulaza. U slučaju komande `odradi`, server odgovara slanjem jedne reči koja opisuje zadatak (npr. 'sudovi'). Ispisati na standardni izlaz poruku:
  `Vas zadatak je: sudovi.` Obrađivati komande sve dok se ne unese komanda `izadji`.
- Implementirati serversku TCP aplikaciju koristeći *Java Socket API*. Server osluškuje na portu 12345, prihvata klijente i vodi računa o kolekciji zadataka. Server takođe loguje klijentske aktivnosti u log fajl `log.txt`. Za svakog klijenta se pokreće posebna nit. (5p)
- Nit koja obrađuje klijenta ima sledeće zadatke: (10p)
  - Čita ime klijenta nakon što se klijent predstavi.
  - Čita komande koje klijent šalje i obrađuje ih.
    U slučaju komande `odradi`, klijentu se iz kolekcije zadataka šalje prvi zadatak, briše se iz kolekcije, i u log se upisuje sledeća poruka:
    `<trenutno_vreme>: Korisnik <ime> je odradio zadatak <trenutni_zadatak>.`
    U slučaju komande `dodaj <rec_koja_opisuje_zadatak>`, zadatak se dodaje u kolekciju, a u log se upisuje sledeća poruka:
    `<trenutno_vreme>: Korisnik <ime> je dodao zadatak <trenutni_zadatak>.`
    U slučaju komande `izadji`, prekida se veza sa klijentom.
- Postarati se da pristup kolekciji zadataka bude sinhronizovan. (1p)
- Postarati se da svi resursi budu pravilno zatvoreni. (1p)

3. **UDP (15p)**
Napisati Java aplikaciju koja će korisnicima omogućiti dohvatanje informacija o delima Leonarda da Vinčija. U direktorijumu `tests` na Desktop-u se nalaze datoteke koje sadrže informacije o Leonardovim delima. Svaka linija je šifrovana tako što je napisana unazad.

- Implementirati Java UDP klijentsku aplikaciju koristeći *Java Datagram API*. Klijent inicira zahtev tako što šalje uzastopno dva datagrama: prvi predstavlja ime fajla a drugi broj linije tog fajla. Server odgovara datagramom koji sadrži **dešifrovanu** liniju. Klijent ispisuje rezultat na standardni izlaz. (5p)
- Implementirati Java aplikaciju koristeći *Java Datagram API* koja predstavlja UDP server. Server treba da, nakon primanja naziva fajla i linije, klijentu vraća dešifrovanu liniju. (5p)
- Postarati se da server radi pravilno iako se redosled datagrama poremeti, drugim rečima ukoliko server treba ispravno da radi iako se prvo od klijenta primi broj linije a zatim ime fajla. (2p)
- Postarati se da server radi pravilno iako se neki od datagrama zagubi — ukoliko od klijenta ne pristignu oba datagrama, ignorisati zahtev. (2p)
- Postarati se da obe aplikacije ispravno upravljaju resursima i pravilno ih otpuštaju u slučaju izuzetka. (1p)