Računarske mreže 4R, Ispit - SEP3, 60p/30p

23.09.2020.

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 sa imenom `rm_rok_Ime_Prezime_miGGXXX` u kome se nalazi validan IntelliJ projekat. Izvući direktorijum iz arhive na Desktop i zameniti svojim podacima. 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/2h**.

1. **Kviz (20p/12p)**
   Napraviti TCP klijent-server aplikaciju preko koje se korisnici takmiče u kvizu. Server ima ulogu sudije u kvizu i vodi evidenciju o poenima takmičara (klijenata).

   - Napraviti Java klasu koja ima ulogu lokalnog TCP servera (koristeći *Java Sockets API*) koji osluškuje na portu 12321. Pri pokretanju, server sa standardnog ulaza učitava nisku koja predstavlja naziv tekstualnog fakla u kome se nalaze pitanja za kviz. Fajl ima format kao u primeru ispod. (2p)
   - Napraviti Java klasu koja ima ulogu TCP klijenta (koristeći *Java Sockets API*). Klijent formira konekciju sa lokanim serverom na portu 12321. Nakon uspostavljanja konekcije, klijent šalje serveru nisku koja predstavlja ime tog klijenta, učitanu sa standardnog ulaza. (2p)
   - Kad se skupi tačno 5 takmičara (klijenata), tada kviz počinje i server šalje svim klijentima poruku *Kviz pocinje sada. Srecno!*. (2p)
   - Za svako učitano pitanje server šalje svim učesnicima kviza poruku u kojoj se nalazi pitanje i 4 ponudjena odgovora. (1p)
   - Klijenti od trenutka kad su primili poruku koja sadrži pitanje, imaju 10 sekundi da pošalju odgovor (A, B, C ili D). (2p)
   - U slučaju da takmičar (klijent):
     (a) zakasni da pošalje odgvor: server mu šalje poruku *Niste stigli da odgovorite na vreme.* i ne menja mu rezultat.
     (b) prvi pogrešno odgovori: server ga obaveštava porukom *Netacan odgovor. Izgubili ste 0.5 poena* i umanjuje rezultat tog klijenta za 0.5 poena.
     (c) prvi tačno odgovori: server mu dodaje jedan poen na trenutni rezultat i obaveštava ga o tome porukom *Tacan odgovor. Osvojili ste poen.*
     (d) nije prvi dao tačan odgovor: server mu šalje poruku *Niste prvi dali tacan odgovor.* i ne menja mu rezultat.

     (5p)
   - Nakon potrošenih svih pitanja u fajlu, server šalje svim klijentima poruku *Kviz je zavrsen!* i rezultate u opadajućem poretku prema broju osvojenih poena u formatu *ImeKlijenta brojPoena* (2p)
   - Postarati se da su svi resursi ispravno zatvoreni u slučaju izuzetka. (1p)

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!

```
ulaz: /home/ispit/Desktop/MaterijaliZaKviz.txt
[Napomena: Sa zvezdicom je odredjen tacan odgovor.]
sadrzaj fajla:
1. Koliko zvezdica je imala zmajeva kugla koju je Goku dobio od deke?
A) 3
B) 4 *
C) 7
D) 5
2. Sta znaci Ichigo na japanskom?
A) jagoda *
B) sargarepa
C) malina
D) paradajz
3. Koju djavolju vocku je pojeo Luffy?
A) Mera Mera no Mi
B) Bari Bari no Mi
C) Toki Toki no Mi
D) Gomu Gomu no Mi *
4. Sta Naruto obozava da jede?
A) Mochi
B) Gyoza
C) Ramen *
D) Sushi
```

2. **UDP (20p/12p)**
Napraviti UDP klijent-server aplikaciju preko koje se šalje deo sadržaja fajla.

- Napraviti Java klasu koja ima ulogu lokalnog UDP servera, koji osluškuje na portu 12121. Server sa standardnog ulaza učitava nisku, koja predstavlja naziv foldera u kome se nalaze fajlovi koje klijenti mogu zatražiti. Implementirati prijem UDP datagrama od klijenata koristeći klasu *DatagramPacket*. Za svaki prihvaćen datagram ispisati poruku na standardni izlaz sa informacijama o tom datagramu — redni broj primljenog datagrama i IP adresu pošiljaoca. (6p/4p)
- Napraviti Java klasu koja ima ulogu UDP klijenta. Klijent sa standardnog ulaza učitava nisku, koja predstvalja naziv fajla i dva neoznačena cela broja koji predstavljaju redne brojeve linija fajla. Klijent šalje serveru datagram, koristeći klasu *DatagramPacket*, koji sadrži naziv fajla i redne brojeve linija. Klijent zatim dobija datagram od servera, čiji sadržaj ispisuje na standardni izlaz. (6p/4p)
- Za svaki prihvaćen datagram sadržaja `naziv, broj1, broj2`, server pošiljaocu odgovara datagramom koji sadrži isečak fajla čiji je naziv određen niskom `naziv` od linije `broj1` do linije `broj2`. Fajlovi se traže samo u okviru odgovarajućeg foldera servera učitanog na početku rada servera. Voditi računa o ispravnim granicama `[broj1, broj2]` i o tome da li fajl sadrži odgovarajući broj linija. U slučaju da nisu ispunjeni ovi uslovi, poslati klijentu poruku `Nepravilno zadata naredba`. (6p/3p)
- Postarati se da su svi resursi ispravno zatvoreni u slučaju izuzetka. (2p/1p)

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!

3. **Tokovi podataka i niti (15p) (za studente koji nisu radili projekat)**
Napisati program koji ispisuje ukupan broj pojavljivanja zadatog karaktera u svim tekstualnim fajlovima sa spiska URL-ova.

- U datoteci `urls.txt` unutar direktorijuma `tests` na Desktop-u se nalazi spisak URL-ova (po jedan u svakoj liniji). Koristeći odgovarajuće **baferisane** ulazne tokove pročitati sadržaj pomenutog fajla i ispisati broj linija u tom fajlu. (2p)
- Za svaku pročitanu liniju fajla `urls.txt` kreirati novi URL objekat koristeći `URL` klasu. Preskočiti sve linije koje ne predstavljaju validan URL. (1p)
- Za svaki validni URL proveriti protokol koji se koristi. Ukoliko je protokol `FILE` i ukoliko putanja vodi do tekstualnog fajla (ekstenzija `.txt`), kreirati zasebnu nit koja će otvoriti **baferisani** ulazni tok do tog fajla putem URL klase i pročitati sadržaj fajla (detalji obrade su u narednoj stavci). Kodnu stranu prilikom učitavanja postaviti na UTF-8. Ukoliko fajl na datoj putanji ne postoji, ispisati poruku i ugasiti nit. (5p)
- Pre parsiranja fajla `urls.txt`, sa standardnog ulaza učitati jedan karakter. Prebrojati koliko se puta zadati karakter pojavljuje u svim fajlovima iz prethodne stavke tako što će svaka nit prebrojati pojavljivanja za fajl koji joj je dodeljen. Ispisati ukupan broj na standardni izlaz (videti primere ispisa ispod teksta zadatka). Pritom, paziti na sinhronizaciju niti ukoliko se koristi deljeni brojač. (5p)
- Postarati se da program ispravno barata specijalnim slučajevima (npr. ako fajl ne postoji na datoj putanji) i ispravno zatvoriti sve korišćene resurse u slučaju izuzetka. (2p)

```
ulaz:  a
izlaz: lines:     29
       not found: /home/ispit/Desktop/tests/404.txt
       result:    3915

ulaz:  %
izlaz: lines:     29
       not found: /home/ispit/Desktop/tests/404.txt
       result:    0

ulaz:  č
izlaz: lines:     29
       not found: /home/ispit/Desktop/tests/404.txt
       result:    1
```