Računarske mreže, Ispit - JAN1 2023

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!**

1. **URL Scanner (15p)**

   - Napraviti Java aplikaciju koja sa standarnog ulaza učitava URL-ove jedan po jedan u svakoj liniji. Formirati URL objekat za svaki učitani URL i ispisati na standardni izlaz poruku ukoliko URL nije validan. (3p)
   - Ispisati protokol, `authority` i putanju iz URL-a koristeći URL klasu. Izlaz formatirati na sledeći način: `<KORIŠĆENI_PROTOKOL> <AUTHORITY> <PUTANJA_DO_RESURSA>` npr.
     ulaz: `http://www.matf.bg.ac.rs:3030/dir1/dir2/test.txt`
     izlaz: `http www.matf.bg.ac.rs:3030 /dir1/dir2/test.txt` (5p)
   - Ukoliko se unese IP adresa unutar URL-a dodatno uz informacije iznad ispisati informaciju o verziji IP adrese koja je uneta i ako je to IPv4 adresa ispisati njene bajtove, u formatu:
     `(v<VERZIJA_IP_ADRESE>) <KORIŠĆENI_PROTOKOL> <PUTANJA_DO_RESURSA> [<BAJTOVI_ADRESE>]` npr.
     ulaz: `http:///123.123.123.123:80/dir1/dir2/test.txt`
     izlaz: `(v4) http /dir1/dir2/test.txt [123 123 123 123]`
     ulaz: `sftp://2001:0db8:85a3:::8a2e:0370:7334/dir1/dir2/test.txt`
     izlaz: `(v6) sftp /dir1/dir2/test.txt` (6p)
   - Postarati se da u slučaju izuzetka aplikacija ispravno zatvori korišćene resurse. (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!

2. **Pogodi broj (TCP) (25p)** Napraviti TCP klijent-server aplikaciju koja igra igru "Pogodi o čemu razmišljam". Server iz fajla `moguce_reci.txt` čita i bira jednu reč nasumično koju klijent treba da pogodi.

   - Implementirati klijentsku TCP aplikaciju koristeći `Java Socket API`. Klijent se povezuje na server na portu `12321`, prima poruku `"Pogodi o cemu razmisljam...."`, kao i sve moguće reči, i šalje reč koju pogađa. Ako nije pogodio reč koju je server zamislio, prima poruku `"Netacno!"`, sa nekim dodatnim infomarcijama o reči (server mu nakon svakog promašaja daje naredno slovo reči koje je zamislio). Ako je pogodio reč, stiže mu poruka `Čestitam! Pogodili ste rec.`. Ako je klijentu prikazano sve osim pretposlednjeg slova reči, i i dalje ne uspe da pogodi, stiže mu poruka `"Nisi uspeo da pogodis... (rec koju je pokusao da pogodi)"`. (7p)
   - Implementirati serversku TCP aplikaciju koristeći *Java Socket API*. Uloga servera je da osluškuje na portu `12321`, prihvata klijente i pokreće zasebnu nit za svakog klijenta. (4p)
   - Pojedinačne niti koje obradjuju klijente imaju ulogu onog koji zamišlja reč. Potrebno je izabrati nasumičnu reč iz fajla `moguce_reci.txt` koju će klijent probati da pogodi. Obavestiće klijenta da je igra počela sa porukom `"Pogodi o cemu razmisljam..."`, kao i ceo izbor mogućih reči (koje su bile pročitane iz fajla). Nakon toga, prima reč koju je klijent poslao, proverava da li je pogodio reč, i šalje mu odgovarajuću poruku. Igra se završava kada klijent pogodi reč, ili kada nakon više poslatih nagoveštaja, klijent i dalje ne pogodi. (11p)
   - Postarati se da su svi resursi ispravno zatvoreni u slučaju naglog isključivanja klijenta ili drugih izuzetaka. (3p)

   Primer rada:

```
server : Pogodi o cemu razmisljam... [slika, sto, pas, prozor, laptop, slusalice,
    stativa, ptica, mesec]
klijent: slika
server : Netacno! (s)
klijent: slusalice
server : Netacno! (st)
klijent: stativa
server : Nisi uspeo da pogodis....(sto)
```

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. **TrendingUDP (20p)**

Implementirati klijent-server aplikaciju koja koristi *Java Datagram API*. U datoteci `trending-songs.txt` nalaze se informacije o preko 30 najpopularnijih pesama na jednoj striming platformi. Pesme su sortirane prema popularnosti. Indeks popularnosti pesme odgovara broju linije iz datoteke (0-indeksirano).

```
Spisak pesama u datoteci 'trending-songs.txt':
  I Feel the Earth Move -- Carole King (2:59)
  What About Me? -- Snarky Puppy (6:42)
  My Oh My -- Leonard Cohen (3:36)
  Senor Mouse -- Gary Burton, Chick Corea (6:17)
  Song For My Father -- Horace Silver (7:18)
  ...
  Ne Me Quitte Pas -- Nina Simone (3:35)
```

- Napisati program koji ima ulogu lokalnog *UDP* servera koji osluškuje na portu *12345*. (3p)
- Implementirati klijentsku stranu kao program nezavisan od servera. Na početku izvršavanja serveru se šalje datagram koji sadrži vrednost *-1*. Klijent treba da čeka na odgovor od servera ne duže od *5s*, a ukoliko odgovor ne stigne, da ponovi slanje istog zahteva. Ukoliko ni tada (nakon 5s) ne stigne odgovor, klijent treba da bude zaustavljen sa ispisom poruke o grešci. (5p)
- Sa serverske strane treba prihvatati datagrame od klijenata, i za svaki, odgovoriti pošiljaocu svojim datagramom koji sadrži pesmu (pročitanu iz fajla) sa traženim indeksom. U slučaju da nema toliko pesama u fajlu (ili je prosleđeno *-1*), server treba da šalje informacije o prvoj narednoj pesmi u odnosu na onu liniju do koje se stiglo u tim slučajevima (globalno posmatrano). Zatim na standardni izlaz treba da ispiše trenutno vreme i informacije o odabranoj pesmi. Ukoliko se stiglo do poslednje pesme u fajlu, a potrebno je odbrati prvu sledeću, postupak nastaviti ciklično – od početka fajla. (5p)
- Nakon prvog primljenog paketa (na klijentskoj strani) treba omogućiti učitavanje sa standardnog ulaza, u beskonačnoj petlji. Očekuje se da svaka linija sadrži ili komandu `next <ind>` (gde je `<ind>` brojčana vrednost koja može biti izostavljena) ili komandu `exit`. Bilo kakvu drugu liniju treba ignorisati (tj. treba čekati na unos naredne linije). Ukoliko se unese `exit`, klijent (uz poruku o završetku) završava sa radom. Ako se sa standardnog ulaza unese `next <ind>`, tada se serveru šalje datagram paket koji sadrži brojčanu vrednost `<ind>`, a ukoliko `<ind>` nije navedeno, datagramom poslati vrednost *-1*. Klijent treba da je u stanju da nakon slanja (a između dva čitanja sa standardnog ulaza) prima odgovor od servera. Kriterijumi čekanja, tj. ponovnog slanja ili obustavljanja rada su isti kao i za prvi datagram. (6p)
- Postarati se da aplikacija vrši obradu grešaka i da ispravno zatvara resurse. (1p)

Napomena: u slučaju da je paket bio zagubljen na putu od servera do klijenta, tj. u slučaju ponovljenog slanja istom klijentu, ne mora se slati ista pesma. Dodatni test primeri nalaze se u `tests` direktorijumu.

```
Primer interakcije sa klijentskim delom programa:
  Sending request...
  Next song: I Feel the Earth Move -- Carole King (2:59)
  >> next 4
  Sending request...
  Next song: Song For My Father -- Horace Silver (7:18)
  >> bad commands will be ignored
  >> just like this and previous one
  >> next
  Sending request...
  Next song: What About Me? -- Snarky Puppy (6:42)
  >> next 0
  Sending request...
  Next song: I Feel the Earth Move -- Carole King (2:59)
  >> next
  Sending request...
  Next song: My Oh My -- Leonard Cohen (3:36)
  exit
  Process finished with exit code 0
```