Računarske mreže, Ispit - JUN1 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!**

1. **URL Scanner (15p) (za studente koji nisu radili projekat)**
   Napisati Java aplikaciju koja obrađuje URL-ove. U direktorijumu `urls`, unutar direktorijuma `tests` na Desktop-u, nalaze se datoteke koje sadrže spisak URL-ova (po jedan u svakoj liniji).

   - Za svaku datoteku pokrenuti zasebnu nit koja će obrađivati i ispisivati statistiku za tu datoteku. (2p)
   - Za svaku pročitanu liniju datoteke kreirati novi URL objekat koristeći `URL` klasu. Preskočiti sve linije koje ne predstavljaju validan URL. (1p)
   - Za svaki validni URL ispisati protokol, `authority` i putanju iz URL-a koristeći URL klasu. Izlaz formatirati na sledeći način:
     `<KORIŠĆENI_PROTOKOL> <AUTHORITY> <PUTANJA_DO_RESURSA>`
     Primer:
     ulaz: `http://www.matf.bg.ac.rs:3030/dir1/dir2/test.txt`
     izlaz: `http www.matf.bg.ac.rs:3030 /dir1/dir2/test.txt` (4p)
   - Postarati se da se ispisi svake niti na standardni izlaz ne prepliću. (3p)
   - Ukoliko se unese IP adresa unutar URL-a, dodatno uz informacije iznad ispisati i 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>]`
     Primer:
     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` (4p)
   - Postarati se da program ispravno obradjuje specijalne slučjeve i ispravno zatvoriti sve korišćene resurse 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!

2. **UDP Sockets (20p/12p)**
Napraviti osnovu za klijent-server Java aplikaciju koristeći `Datagram API` koja šifruje poruku koristeći Morzeov kôd. Morzeova azbuka se sastoji od dva simbola: `.` (*dit*) i `-` (*dah*). Svaki karakter se kodira odgovarajućim nizom ovih simbola, a čitave poruke se kodiraju tako što se dodaje pauza nakon svakog karaktera, kao i duža pauza nakon svake reči.

- Napraviti Java aplikaciju koja ima ulogu UDP klijenta. Poslati datagram lokalnom serveru na portu 23456 koja sadrži nisku (koja može da sadrži beline). Niska se učitava sa standardnog ulaza na klijentskoj strani. Primiti datagram koji predstavlja odgovor servera i rezultat ispisati na standardni izlaz. (8p/4p)
- Napraviti Java aplikaciju koja ima ulogu servera koji sluša na portu 23456. Kada dobije nisku od klijenta tu nisku transformiše Morzeovom azbukom u šifrovanu poruku koristeći datoteku *morse.txt*. Posle svakog karaktera dodati razmak koji predstavlja pauzu, a nakon svake reči dodati tri razmaka koji predstavljaju trostruku pauzu. Na kraju poruke dodati signal za kraj: `.-.-.-` (10p/7p)
- Postarati se da su svi resursi ispravno zatvoreni u slučaju izuzetka. (2p/1p)

```
Klijent:   Morze
Server:    -- --- .-. --.. .   .-.-.-

Klijent:   Orao Pao
Server:    --- .-. .- ---   .--. .- ---   .-.-.-
```

3. **Non-Blocking IO (25p/18p)**
Napraviti klient-server Java aplikaciju koristeći `TCP Sockets/Channels API` koja se može iskoristiti kao osnova za kreiranje kartaških igara.

- Napisati Java klasu koja ima ulogu **blokirajućeg** TCP klijenta koristeći `Java Channels API`. Klijent formira konekciju sa lokanim serverom na portu 12345 i zatim šalje serveru broj učitan sa standardnog ulaza koji predstavlja broj karti koje se izvlače iz serverskog špila. (4p/3p)
- Napisati Java klasu koja ima ulogu lokalnog **neblokirajućeg** TCP servera, koji osluškuje na portu 12345, koristeći `Java Channels API`. Karte predstaviti na *proizvoljan* način, sa narednim ograničenjima:
  - Svaka karta se sastoji od vrednosti i znaka
  - Vrednost karte može biti od 2 do 10 za karte bez slike dok žandar ima vrednost 11, kraljica 12, kralj 13 i kec 14
  - Znak može biti `pik`, `herc`, `tref` ili `karo`

  (2p/2p)
- Server slučajnim izborom generiše jean špil karata prilikom pokretanja. Kreirati standardni špil od 52 karte i promešati ga prilikom kreiranja servera. Nakon toga ispisati špil na standardni izlaz, svaku kartu u zasebnom redu, u formatu `vrednost.znak`, na primer `4.Pik` ili `14.Karo`. (3p/2p)
- Nakon konekcije, klijent šalje serveru broj karti koje želi da izvuče iz špila. Server izvlači toliko karti sa vrha špila šalje ih klijentu (proizvoljno implementirati slanje). Jednom izvučene karte se ne vraćaju u špil. Klijent ispisuje primljene karte na standardni izlaz. (10p/8p)
- Ukoliko je broj koji je klijent poslao veći od trenutnog broja karata u špilu ili manji od 1, poslati tu informaciju korisniku na proizvoljan način i ispisati odgovarajuću poruku na klijentskoj strani. (4p/2p)
- Postarati se da su svi resursi ispravno zatvoreni u slučaju izuzetka. (2p/1p)