Računarske mreže, Ispit - SEP2 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 IntelliJ projekat u formatu `rm_rok_Ime_Prezime_mXGGXXX`.
Dekompresovati arhivu na Desktop i ubaciti svoje podatke u ime pomenutog direktorijuma.
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. **TCP Sockets (25p/18p)**
   Napraviti osnovu za TCP klijent-server Java aplikaciju koja pomaže u održavanju server farme. Održavanje server farme podrazumeva obilazak i obradu svakog računara. Takodje, u slučaju kvara, neophodno je ispraviti odgovarajući računar. Pošto je broj računara u farmi veliki, potrebno je puno tehničara i sistem za praćenje trenutnog stanja farme. Sistem bi trebalo da čuva status za svaki od računara u mrežnoj topologiji farme i da omogući tehničarima da udaljeno promene statuse računara. Odlučeno je da se za ove potrebe kreira Java serverska aplikacija koja će omogućiti tehničarima da se povežu i postavljaju upite sistemu — kakvo je trenutno stanje farme, markiraj računar `X` kao pokvaren, itd. Topologija serverske farme je takva da se može predstaviti matricom veličine `NxN` (izgledno je da će se veličina menjati u budućnosti) i da se svaki računar može "adresirati"parom brojeva koji predstavljaju vrstu i kolonu u matrici, redom.

   - Napraviti Java klasu koja ima ulogu lokalnog TCP servera koji osluškuje na portu 13370. Pre pokretanja servera, zatražiti od korisnika da unese broj `N` koji predstavlja veličinu matrice server farme. Nakon toga, server osluškuje na pomenutom portu i svakom novom klijentu inicijalno šalje stanje serverske farme — trenutno stanje matrice farme. Ispisati matricu kao u primeru ispod — sa `o` su označeni ispravni, a sa `x` neispravni računari. Prilikom pokretanja servera pretpostaviti da su svi računari ispravni. Server svakog klijenta treba da obradi u zasebnoj niti. (4p/2p)
   - Napraviti Java klasu koja ima ulogu lokalnog TCP klijenta. Nakon uspostavljanja konekcije sa serverom na portu 12345, klijent čeka na trenutno stanje farme od servera i ispisuje ga na standardni izlaz. Nakon toga, klijent šalje serveru upite koje tehničar unosi sa standardnog ulaza, sve dok se ne unese upit `exit`, nakon čega klijent prestaje s radom. (3p/2p)
   - Nakon slanja stanja farme klijentu, server čeka na upite klijenta. Upiti mogu biti: (13/11p)
     (a) `list` — odgovor servera treba da bude trenutno stanje farme
     (b) `mark X Y` — markira server sa koordinatama `(X, Y)` kao neispravan i šalje klijentu ažurirano stanje farme
     (c) `repair X Y` — uklanja marker neispravnosti serveru sa koordinatama `(X, Y)` i šalje klijentu ažurirano stanje farme
     (d) `exit` — server prestaje da obradjuje klijenta i raskida vezu (bez slanja odgovora klijentu)
     (ukoliko koordinate nisu ispravne, server ne vrši nikakve modifikacije)
   - Sinhronizovati pristup modelu farme na serverskoj strani. (3p/2p)
   - Postarati se da su svi resursi ispravno zatvoreni u slučaju izuzetka. (2p/1p)

```
$ java Client.java
+|01234
++-----
0|ooooo
1|ooooo
2|ooooo
3|ooooo
4|ooooo
```

```
(upit) mark 3 3
+|01234
++-----
0|ooooo
1|ooooo
2|ooxoo // markirano polje (3,3)
3|oooox // drugi tehnicar markirao
4|ooooo
```

```
(upit) repair 3 3
+|01234
++-----
0|oooxo
1|ooooo
2|ooooo // ispravljeno
3|oooox
4|ooooo
```

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 - Cezarova šifra (20p/12p)**
   Napraviti klijent-server Java aplikaciju, koristeći `Datagram API`, koja će služiti za šifrovanje poruka.

   - Napisati Java klasu koja ima ulogu lokalnog UDP klijenta. Klijent sa standardnog ulaza učitava rečenice koje želi da šifruje, sve dok ne unese *KRAJ*. Za svaku učitanu rečenicu, klijent šalje serveru paket sa rečenicom kao sadržajem. Za svaki poslati paket, klijent dobija paket nazad od servera, čiji sadržaj ispisuje na standardni izlaz. (6p/4p)
   - Napisati Java klasu koja ima ulogu lokalnog UDP servera koji osluškuje na portu 12345. Server prihvata pakete od klijenata i za svaki prihvaćen paket ispisuje poruku na standardni izlaz sa informacijama o rednom broju primljenog paketa i portu sa kog je paket stigao. (6p/3p)
   - Za svaki prihvaćen paket, server šifruje rešenicu i šalje je nazad klijentu. Najpre slučajnim izborom generiše ključ, jedan ceo broj u intervalu [0, 25], koji ispisuje na standardni izlaz, a zatim svaki karakter rečenice zamenjuje karakterom iz alfabeta ciklično pomerenim za broj mesta određenim vrednošću ključa. Na primer, za vrednost ključa jednak 3, a će biti zamenjeno karakterom d, a z karakterom c. Pretpostaviti da će sve rečenice biti napisane svim malim slovima. (6p/4p)
   - Postarati se da su svi resursi ispravno zatvoreni u slučaju izuzetka. (2p/1p)

3. **Nesto (15p) (za studente koji nisu radili projekat)**