

Računarske mreže, Ispit - SEP1 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_mXGXXXX`.
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. Takođe, 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 status računara. Odlučeno je da se za ove potrebe kreira Java server-ska 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 $N \times N$ (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)

<pre>\$ java Client.java (upit) mark 3 3 + 01234 ++----- 0 ooooo 1 ooooo 2 ooooo 3 ooooo 4 ooooo</pre>	<pre>(upit) mark 3 3 + 01234 ++----- 0 ooooo 1 ooooo 2 ooxoo // markirano polje (3,3) 3 ooxox // drugi tehnicar markirao 4 ooooo</pre>	<pre>(upit) repair 3 3 + 01234 ++----- 0 oooxo 1 ooooo 2 ooooo // ispravljeno 3 ooxox 4 ooooo</pre>
--	--	---

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

2. Protocol handlers (20p/12p)

Implementirati podršku za URL-ove koji koriste `farm` protokol. Opis protokola je dat u prethodnom zadatku, pretpostaviti da je server pokrenut na istom portu kao i u prethodnom zadatku.

- Prilikom otvaranja konekcije, formirati vezu koristeći `Socket API`. Povezati se na server i port na osnovu URL-a i otvoriti ulazni tok do odgovora od strane servera. (5p/3p)
- Omogućiti slanje upita pomoću parametra `oblast` iz URL-a, npr. za upit `repair 3 4`, kompletan URL bi bio:

```
farm://localhost:13370?q=repair&x=3&y=4
```

Server šalje nazad odgovor u primeru iznad (jedno stanje farme prilikom ostvarivanja konekcije, tj. pre upita, i jedno stanje farme nakon upita). (5p/3p)

- Ukoliko port nije naveden unutar URL-a, iskoristiti predefinisani podrazumevani port isti kao u prethodnom zadatku. (1p/1p)
- Predefinisati `getInputStream()` metod da vraća ulazni tok do odgovora od strane servera ukoliko je konekcija ostvorena, a `null` ako nije. (1p/1p)
- Postarati se da je moguće bezbedno koristiti implementirani handler u višenitnom okruženju. (1p/1p)
- Napisati jednostavan test - kreirati URL, otvoriti konekciju do resursa i ispisati sve podatke koje server pošalje. (2p/1p)

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

Napisati program koji na standardni izlaz ispisuje informacije o matricama.

- U direktorijumu `fajlovi` unutar direktorijuma `tests` na Desktop-u se nalaze datoteke koje sadrže matrice (prva dva broja odgovaraju dimenzijama matrice nakon čega slede elementi matrice). Na standardni izlaz, za svaku datoteku sa ekstenzijom `.txt`, ispisati naziv datoteke i da li je matrica gornje trougaona ili ne (ispisati *da* ili *ne*). Matrica je gornje trougaona ako su elementi ispod dijagonale jednaki 0. (6p)
- Za svaku datoteku pokrenuti zasebnu nit koja će obrađivati i ispisivati informacije. (3p)
- Postarati se da se ispisi svake niti na standardni izlaz ne prepliću. (4p)
- Postarati se da program ispravno obrađuje specijalnim slučajeve (npr. ako datoteka ne postoji na datom putanji) i ispravno zatvoriti sve korišćene resurse u slučaju izuzetka. (2p)