Računarske mreže 4R, Ispit - SEP2, 60p/30p

09.09.2020.

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. **Non-Blocking IO (25p/18p)**
   Napraviti osnovu za klijent-server Java aplikaciju koja izvršava bankovske transakcije. Prilikom povezivanja na server klijenti dobijaju spisak brojeva računa svih trenutno povezanih klijenata. Podrazumevano, svi klijenti su primaoci sve dok ne odaberu broj računa na koji žele da pošalju novac. U tom trenutku klijent postaje pošiljalac i server očekuje od njega iznos koji želi da prebaci na odredišni račun. U ovoj inicijalnoj verziji aplikacije se ne očekuje potpuna implementacija korisničkih računa već se podrazumeva da svaki transfer uspeva.

   - Napraviti Java aplikaciju koja ima ulogu klijenta. Povezati se na lokalni server na portu 12221 koristeći **blokirajući Java Channels API**. Nakon formiranja konekcije klijent serveru šalje svoj broj računa a zatim očekuje od servera spisak brojeva računa svih trenutno povezanih klijenata (po jedan u svakom redu) i ispisuje isti na standardni izlaz. Nakon toga, klijent čeka na unos broja računa primaoca sa standardnog ulaza (broj računa ne mora da se unese) i šalje ga serveru zajedno sa iznosom koji predstavlja neoznačeni ceo broj takođe učitan sa standarnog ulaza. Nakon slanja ovih podataka, server odgovara podrvrdom o izvršenju transakcije i klijent ispisuje odgovor na standardni izlaz. Primalac takođe prima poruku o izvršenju transakcije i istu ispisuje na standardni izlaz. (7p/5p)
   - Napraviti Java aplikaciju koja ima ulogu servera. Otvoriti lokalni server na portu 12221 koristeći **neblokirajući Java Channels API**. Nakon primanja klijenta, server od njega očekuje broj računa i nakon primanja broja računa mu šalje brojeve računa svih trenutno povezanih klijenata. Nakon toga, server čeka na poruke od klijenata. Oni klijenti koji pošalju broj računa primaoca se tumače kao pošiljaoci i od njih se dodatno očekuje i iznos koji žele da prebace na račun primaoca. Server svakom pošiljaocu šalje nisku `"Transfer na racun BROJ_RACUNA (IZNOS) je uspesno izvrsen"`, a svakom primaocu šalje nisku `"Primljeno IZNOS od BROJ_RACUNA"` (**nije neophodno implementirati stanje računa na serverskoj strani, pretpostaviti da svaki transfer uspeva**). (13p/10p)
   - Obezbediti da u slučaju izuzetaka, svi resursi budu ispravno zatvoreni i da se pošiljaocu pošalje poruka `Primalac nije pronadjen` ukoliko je primalac prekinuo vezu pre iniciranja transfera. (5p/3p)

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)**
Implementirati UDP server koji korisnicima daje uvid u stanje njihovog računa. Računi mogu učitati tokom rada servera i server je dužan da te izmene primeni u toku svog izvršavanja. Klijenti slanjem datagrama serveru sa svojim brojem računa dobijaju uvid u trenutno stanje računa.

- Napraviti Java aplikaciju koja ima ulogu UDP klijenta. Poslati inicijalni datagram lokalnom serveru na portu 12345 sa brojem računa. Primiti datagram od servera koji sadrži iznos koji se nalazi na računu i ispisati iznos na standardni izlaz (informacije proizvoljno kodirati u datagrame). Iznos posmatrati kao pozitivan realan broj. (5p/2p)
- Napraviti Java aplikaciju koja ima ulogu UDP servera.
  - Kreirati nit za unos novih računa sa standardnog ulaza. Tokom rada servera, korisnik može da unese nove brojeve računa u formatu `BROJ_RACUNA IZNOS`. Server je dužan da taj račun doda u spisak aktivnih računa. Iznos posmatrati kao pozitivan realan broj. Pri pokretanju servera smatrati da ne postoji nijedan registrovan aktivni račun. (8p/5p)
  - Kreirati nit za slanje/primanje datagrama. Kad god server primi datagram od klijenta, izvlači broj računa i odgovara datagramom koji sadrži informaciju o iznosu koji je trenutno dodeljen tom računu (informacije proizvoljno kodirati u datagrame). Ukoliko broj računa nije aktivan, poslati `-1` kao trenutno stanje. (6p/4p)
- Postarati se da su svi resursi ispravno zatvoreni u slučaju izuzetka. (1p/1p)

3. **Sortiranje (15p) (za studente koji nisu radili projekat)**
Napraviti Java aplikaciju koja pomoću niti sortira kolekciju brojeva.

- U fajlu, čije se ime zadaje preko standardnog ulaza, se nalazi kolekcija čiji su elementi realni brojevi. Elementi kolekcije su u fajlu razdvojeni belinama. Učitati kolekciju i na standardni izlaz ispisati ukupan broj elemenata kolekcije. (3p)
- Za svaki element kolekcije pokrenuti zasebnu nit koja će vratiti njegov *indeks* — broj elemenata kolekcije koji su pre njega u sortiranom poretku. (6p)
- Formirati strukturu podataka za sortiranje, koja će na osnovu elementa kolekcije i njegovog indeksa čuvati elemente u sortiranom poretku. Imati u vidu da se isti broj može javiti više puta. (3p)
- Na standardni izlaz ispisati sortiranu kolekciju. (1p)
- Na standardni izlaz ispisati koliko je vremena bilo potrebno da se kolekcija sortira u milisekundama. (1p)
- Postarati da su svi resursi ispravno zatvoreni u slučaju izuzetaka (1p)