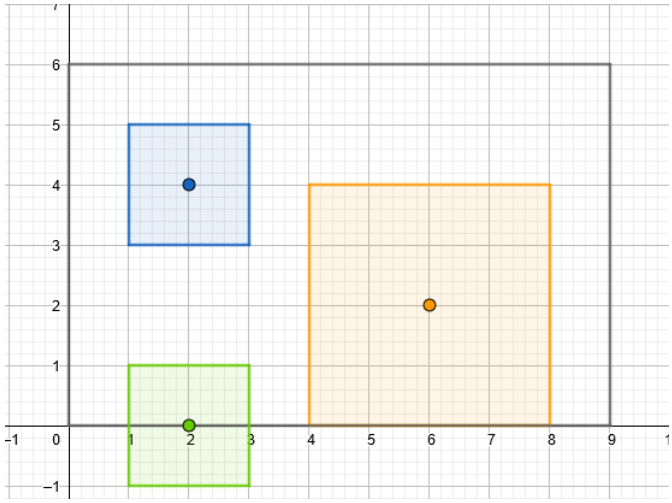
Računarske mreže 4R, Ispit - SEP1, 60p/30p

26.08.2020.

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)**
   Za potrebe skeniranja terena odlučeno je da se kreira *hab* (server) koji će prikupljati podatke koji se dobijaju od *skenera* (klijenata). Svaki skener se odlikuje svojom pozicijom (x, y) i može da pokrije odgovarajuću oblast radijusa r. Cilj je pokriti čitav teren skenerima. Teren se posmatra kao mreža jediničnih kvadrata dužine m i širine n (videti sliku ispod), dok je (x, y) jedno teme mreže. Skener pokriva kvadrat veličine $2r$ sa centrom u (x, y). m, n, x, y i r su nenegativni celi brojevi.



Slika 1: Teren dimenzija 9×6 sa tri postavljena skenera. Pokrivenost u ovom slučaju: $\frac{22}{9 \times 6} \approx 0.407 = 40.7\%$

- Napraviti Java aplikaciju koja ima ulogu skenera. Povezati se na lokalni server na portu 7337 koristeći **blokirajući Java Channels API**. Nakon formiranja konekcije, klijent serveru šalje brojeve (x, y) i r koje operater skenera unosi sa standardnog ulaza. Nakon slanja, klijent ispisuje odgovore od servera sve dok server ne prekine vezu. (6p/4p)
- Napraviti Java aplikaciju koja ima ulogu centralnog haba. Pokrenuti lokalni server na portu 7337, koristeći **neblokirajući Java Channels API**. Prilikom pokretanja, operater haba unosi veličinu terena — brojeve m i n. Hab zatim opslužuje skenere. Nakon uspostavljanja veze, hab prima podatke od skenera — (x, y) i r (pozicija skenera mora biti unutar granica terena, u protivnom se raskida veza sa tim skenerom). U međuvremenu, na svakih 5 sekundi, hab svim skenerima šalje trenutnu pokrivenost terena — procenat jediničnih kvadrata koji su pokriveni skenerima u odnosu na ukupan broj jediničnih kvadrata u mreži. (13p/9p)
- Server raskida vezu sa klijentima i završava sa radom onda kada pokrivenost terena postane 100%. (3p/2p)
- Obezbediti da u slučaju izuzetaka, svi resursi budu ispravno zatvoreni i da se ukupna pokrivenost terena eventualno promeni ukoliko je neki skener prekinuo vezu! (3p/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 na osnovu informacija o terenu i postavljenim skenerima klijentima šalje odgovor o pokrivenosti njihove lokacije (temena mreže terena).

   - Napraviti Java aplikaciju koja ima ulogu UDP klijenta. Poslati inicijalni datagram lokalnom serveru na portu 12345 sa lokacijom klijenta na mreži terena (videti sliku sa prethodne strane). Ispisati `Pokriven!` ako je server odgovorio pozitivno na datagram (lokacija je pokrivena bar jednim skenerom) ili `Nije pokriven!` ukoliko je server odgovorio negativno na datagram (informacije proizvoljno kodirati u datagrame). (5p/3p)
   - Napraviti Java aplikaciju koja ima ulogu UDP servera. Najpre iz fajla `terrain.txt` učitati podatke o terenu i postavljenim skenerima. Primer fajla je dat ispod. (3p/2p)
   - Nakon učitavanja podataka o terenu, slušati na portu 12345 i ispisati tekst `Pristigao klijent!` kad god server primi datagram. Odgovoriti datagramom koji sadrži informaciju o pokrivenosti lokacije izvučene iz datagrama — lokacija je pokrivena ako je u radijusu barem jednog skenera (informacije proizvoljno kodirati u datagrame). (10p/6p)
   - Postarati se da su svi resursi ispravno zatvoreni u slučaju izuzetka. (2p/1p)

```
9 6
3 4 1
2 0 1
6 2 2
```

Slika 2: Primer fajla `terrain.txt` koji odgovara slici sa prethodne strane — u prvoj liniji se nalaze dimenzije terena, a zatim u narednim linijama se nalaze informacije o postavljenim skenerima, po jedan skener u svakoj liniji (redosled nije bitan, u ovom primeru je redosled: plavi, zeleni pa crveni).

Primer izvršavanja:

```
terrain.txt ucitan!
Server pokrenut!
                      Klijent pokrenut!
                      > 2 1
Stigao datagram!
                      Pokriven!
                      Klijent pokrenut!
                      > 2 2
Stigao datagram!
                      Nije Pokriven!
                      Klijent pokrenut!
                      > 7 3
Stigao datagram!
                      Pokriven!
                      Klijent pokrenut!
                      > 100 100
Stigao datagram!
                      Nije Pokriven!
```

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. **Frekvencija reči (15p) (za studente koji nisu radili projekat)**
   Napraviti Java aplikaciju koja pomoću niti broji pojavljivanja svih reči u okviru tekstualnih fajlova unutar zadatog direktorijuma.

   - Kao ulaz u program se daje putanja do direktorijuma (u kome može biti više poddirektorijuma) u kojem se nalaze tekstualni fajlovi. Na standardni izlaz ispisati putanju do svakog regularnog fajla sa ekstenzijom `.txt` unutar tog direktorijuma kao i broj linija u svakom od pomenutih fajlova. (5p)
   - Napraviti zajedničku stukturu podataka u kojoj će se voditi evidencija o broju pojavljivanja reči u svim tekstualnim fajlovima unutar zadatog direktorijuma. (1p)
   - Za svaki pomenuti tekstualni fajl, nakon ispisa broja linija u tom fajlu, pokrenuti zasebnu nit koja će, koriseći odgovarajuće ulazne tokove podataka, da pročita sadržaj tog fajla i ažurira broj pojavljivanja odgovarajućih reči u zajedničkoj strukturi podataka. (3p)
   - Postarati se da nema konfliktnih situacija prilikom ažuriranja zajedničke strukture podataka. (3p)
   - Na standardni izlaz ispisati sadržaj zajedničke strukture podataka sortiran opadajuće po broju pojavljivanja u formatu `rec: brojPojavljivanja`. (2p)
   - Postarati da su svi resursi pravilno zatvoreni u slučaju izuzetka. (1p)