

Računarske mreže 4I, Ispit - SEP1

26.08.2020.

Pročitati sve zadatke **pažljivo** pre rada - sve što nije navedeno ne mora da se implementira!

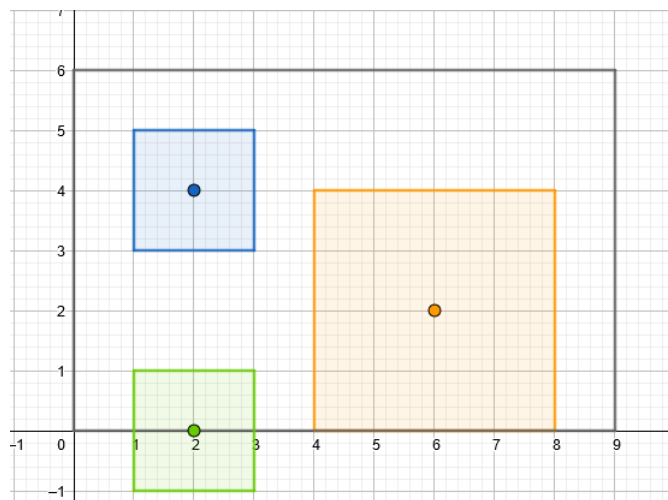
Na **Desktop**-u se nalazi zip arhiva. Unutar arhive se nalazi direktorijum u formatu `rm_rok_Ime_Prezime_miGGXXX` u kome se nalazi validan IntelliJ projekat. Izvući direktorijum iz arhive na Desktop i ubaciti svoje podatke u ime. Otvoriti IntelliJ IDEA, izabrati opciju **Open project** (ne **Import 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! Vreme za rad: **2.5h**.

1. Non-Blocking IO (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 operator skenera unosi sa standardnog ulaza. Nakon slanja, klijent ispisuje odgovore od servera sve dok server ne prekine vezu. (4p)
- Napraviti Java aplikaciju koja ima ulogu centralnog haba. Pokrenuti lokalni server na portu 7337, koristeći **neblokirajući Java Channels API**. Prilikom pokretanja, operator 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. (9p)
- Server raskida vezu sa klijentima i završava sa radom onda kada pokrivenost terena postane 100%. (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)

*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. UDP Sockets (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). (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. (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). (6p)
- Postarati se da su svi resursi ispravno zatvoreni u slučaju izuzetka. (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!

Stigao datagram!

Stigao datagram!

Stigao datagram!

Stigao datagram!

Klijent pokrenut!
> 2 1

Pokriven!
Klijent pokrenut!
> 2 2

Nije Pokriven!
Klijent pokrenut!
> 7 3

Pokriven!
Klijent pokrenut!
> 100 100

Nije Pokriven!
```