Računarske mreže, Ispit - JAN1

18.01.2021.

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 direktorijum u formatu `rm_rok_Ime_Prezime_mXGGXXX` 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: **3h**.

1. **Da Vinčijev URL (20p)**
   Napisati program koji na standardni izlaz ispisuje informacije o skrivenim URL-ovima. U direktorijumu tests na Desktop-u nalaze se datoteke koje sadrze informacije o delima Leonarda da Vinčija — svaki red je jedna informacija. Svaka datoteka osim informacija sadrži i redove koji predstavljaju validane URL-ove. Ali kako je Leonardo poznat po svojim šiframa datoteke se sastoje od linija takvih da je svaka linija ponaosob napisana "unazad"(zdesna nalevo), pa je linije prvo potrebno "dešifrovati".

   - Obići direktorijum `tests` i za svaku datoteku pokrenuti zasebnu nit koja će obraditi tu datoteku (3p)
   - Za liniju koja "dešifrovana"predstavlja validan URL kreirati novi URL objekat koristeći URL klasu, ostale linije preskočiti (2p)
   - Za svaku datoteku ispisati redom sledeće informacije
     - naziv datoteke (1p)
     - za svaki **validni** URL ispisati broj linije u kojoj je pronađen i protokol koji se koristi (format ispisa: `broj_linije : protokol`) (5p)
     - ako je protokol `https` ispisati i port (2p)
     - koliko linija datoteke predstavlja validan URL (1p)

     (Pogledati primer ispisa ispod)
   - Postarati se da se ispisi svake niti na standardni izlaz ne prepliću (4p)
   - Postarati se da program ispravno obrađuje specijalne slučajeve i ispravno zatvara sve korišćene resurse (2p)

```
Spisak datoteka u direktorijumu tests:
    \home\ispit\Desktop\tests\LadyWithAnErmine.txt
    \home\ispit\Desktop\tests\LastSupper.txt
    \home\ispit\Desktop\tests\MonaLisa.txt
    \home\ispit\Desktop\tests\VitruvianMan.txt

Primer ispisa za LastSupper.txt:
    LastSupper
    5 : file
    8 : https : -1
    2
```

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. **Odrad.io (TCP) (25p)**
Zajednički život i održavanje domaćinstva sa sobom donosi razne frustracije, pogotovo u periodu ispitnog roka. Srećom, u frižideru Vas čeka parče omiljene torte koje planirate da pojedete na pauzi spremanja ispita iz Računarskih mreža. Međutim, shvatate da nema čistih kašika, jer Vaš cimer opet nije oprao sudove! Inspirisani novostečenim znanjem iz RM, odlučujete da napravite aplikaciju za praćenje kućnih poslova.

- Implementirati klijentsku TCP aplikaciju koristeći *Java Socket API*. Klijent se povezuje na server na portu 12345 i predstavlja se slanjem svog imena (jedna reč) koje se učitava sa standardnog ulaza. Nakon toga, klijent može poslati neku od tri komande: (8p)
  - `odradi`
  - `dodaj <rec_koja_opisuje_zadatak>`
  - `izadji`

  Komanda se učitava sa standardnog ulaza. U slučaju komande `odradi`, server odgovara slanjem jedne reči koja opisuje zadatak (npr. 'sudovi'). Ispisati na standardni izlaz poruku:
  `Vas zadatak je: sudovi.` Obrađivati komande sve dok se ne unese komanda `izadji`.
- Implementirati serversku TCP aplikaciju koristeći *Java Socket API*. Server osluškuje na portu 12345, prihvata klijente i vodi računa o kolekciji zadataka. Server takođe loguje klijentske aktivnosti u log fajl `log.txt`. Za svakog klijenta se pokreće posebna nit. (5p)
- Nit koja obrađuje klijenta ima sledeće zadatke: (10p)
  - Čita ime klijenta nakon što se klijent predstavi.
  - Čita komande koje klijent šalje i obrađuje ih.
    U slučaju komande `odradi`, klijentu se iz kolekcije zadataka šalje prvi zadatak, briše se iz kolekcije, i u log se upisuje sledeća poruka:
    `<trenutno_vreme>: Korisnik <ime> je odradio zadatak <trenutni_zadatak>.`
    U slučaju komande `dodaj <rec_koja_opisuje_zadatak>`, zadatak se dodaje u kolekciju, a u log se upisuje sledeća poruka:
    `<trenutno_vreme>: Korisnik <ime> je dodao zadatak <trenutni_zadatak>.`
    U slučaju komande `izadji`, prekida se veza sa klijentom.
- Postarati se da pristup kolekciji zadataka bude sinhronizovan. (1p)
- Postarati se da svi resursi budu pravilno zatvoreni. (1p)

3. **UDP (15p)**
Napisati Java aplikaciju koja će korisnicima omogućiti dohvatanje informacija o delima Leonarda da Vinčija. U direktorijumu `tests` na Desktop-u se nalaze datoteke koje sadrže informacije o Leonardovim delima. Svaka linija je šifrovana tako što je napisana unazad.

- Implementirati Java UDP klijentsku aplikaciju koristeći *Java Datagram API*. Klijent inicira zahtev tako što šalje uzastopno dva datagrama: prvi predstavlja ime fajla a drugi broj linije tog fajla. Server odgovara datagramom koji sadrži **dešifrovanu** liniju. Klijent ispisuje rezultat na standardni izlaz. (5p)
- Implementirati Java aplikaciju koristeći *Java Datagram API* koja predstavlja UDP server. Server treba da, nakon primanja naziva fajla i linije, klijentu vraća dešifrovanu liniju. (5p)
- Postarati se da server radi pravilno iako se redosled datagrama poremeti, drugim rečima ukoliko server treba ispravno da radi iako se prvo od klijenta primi broj linije a zatim ime fajla. (2p)
- Postarati se da server radi pravilno iako se neki od datagrama zagubi — ukoliko od klijenta ne pristignu oba datagrama, ignorisati zahtev. (2p)
- Postarati se da obe aplikacije ispravno upravljaju resursima i pravilno ih otpuštaju u slučaju izuzetka. (1p)

Računarske mreže, Ispit - JAN2

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_mXGGXXX` 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!**

1. **CodeSearch (20p)**
   U direktorijumu `tests` na Desktopu, nalazi se kod za Vašu aplikaciju na stručnom kursu kompanije Desni8. Potrebno je napraviti Java aplikaciju koja obilazi direktorijum i pretražuje datoteke. Na standardni izlaz ispisuje sva pojavljivanja tokena koji se unosi sa standardnog ulaza.

   - Rekurzivno obići direktorijum `tests` i za svaku datoteku pokrenuti zasebnu nit koja je pretražuje. (7p)
   - Proći kroz datoteku i ispisati sva pojavljivanja tokena, u sledećem formatu (videti ispod primer): (4p)

     ```
     <PUTANJA DO FAJLA>:<BROJ LINIJE>:<INDEKS POCETKA TOKENA>: <LINIJA>
     ```

   - Voditi računa o tome da su neke od datoteka slike! Datoteke sa ekstenzijama `png`, `ico` ili `svg` ne treba obrađivati. (2p)
   - Postarati se da se ispisi svake niti na standardni izlaz ne prepliću. (5p)
   - Postarati se da su svi resursi pravilno zatvoreni. (2p)

   ```
   Unesite token koji zelite da nadjete:
   div

   tests/react-highscores/src/App.js:7:5:     <div className="App">
   tests/react-highscores/public/index.html:31:5:     <div id="root"></div>
   tests/react-highscores/public/index.html:31:21:     <div id="root"></div>
   tests/front/snake.html:9:5:     <div id="container">
   tests/react-highscores/src/Highscores.js:34:7:       <div className="container">
   tests/front/snake.html:11:7:       <div id="score">0</div>
   tests/front/snake.html:11:25:       <div id="score">0</div>
   tests/react-highscores/src/Highscores.js:52:8:       </div>
   tests/front/snake.html:13:6:     </div>
   tests/react-highscores/src/App.js:9:6:     </div>
   ```

---

*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. **Menjačnica NBIO (25p)**
Napraviti klient-server Java aplikaciju koristeći TCP Sockets/Channels API koja ima ulogu menjačnice.

- Napisati Java klasu koja ima ulogu **blokirajućeg** TCP klijenta koristeći `Java Channels API`. Klijent formira konekciju sa lokalnim serverom na portu `12345`. Nakon formiranja konekcije, klijent šalje serveru zahteve za konverziju, red po red, sve do unosa niske `"prekid"`. Zahtevi su oblika: `valuta iznos`, gde `valuta` predstavlja koju valutu želimo da konvertujemo u dinare, a `iznos` količinu novca koju želimo da konvertujemo (pogledati primere). Pre učitavanja narednog reda potrebno je sačekati odgovor servera i ispisati ga na standardni izlaz. Odgovor servera predstavlja konvertovani iznos u dinarima. (8p)
- Napisati Java klasu koja ima ulogu **neblokirajućeg** TCP servera koji osluškuje na portu `12345` koristeći `Java Channels API`. Server pri pokretanju iz fajla `kursna_lista.txt` čita vrednosti stranih valuta u odnosu na dinar. Nakon konekcije, klijent šalje serveru valute i iznose koje želi da konvertuje u dinare. Kao odgovor, nakon svakog primljenog zahteva, server šalje konvertovan iznos. Konverzija je moguća ako se primljena valuta nalazi u fajlu `kursna_lista.txt`. Pretpostaviti da server ima dovoljan iznos novca kojim može da odgovori na zahteve svih klijenata. (13p)
- Ako server ne zna vrednost tražene valute potrebno je na klijentskoj strani ispisati da nije moguće menjanje te valute. (videti primere ispod). (2p)
- Ako klijent unese negativan broj potrebno je na klijentskoj strani ispisati da iznos novca ne može biti negativan (videti primere ispod). (1p)
- Postarati se da svi resursi budu pravilno zatvoreni. (1p)

Primer rada:

```
ulaz: EUR 5
izlaz: 587.915

ulaz: TRY 4
izlaz: Ne menjamo trazenu valutu

ulaz: pln -7
izlaz: Iznos novca ne moze biti negativan broj

ulaz: hrk 10
izlaz: 156.28
```

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. **Protocol handlers (15p)**
Implementirati podršku za URL-ove koji koriste `exchange` protokol. Opis protokola je dat 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)
- Omogućiti slanje upita pomoću parametara `valuta` i `iznos` iz URL-a, npr. za upit , kompletan URL bi bio:

  ```
  exchange://localhost:1337?valuta=EUR&iznos=5
  ```

  Server šalje nazad rezultat koji klijent ispisuje kao u primeru ispod. (5p)
- Ukoliko port nije naveden unutar URL-a, iskoristiti predefinisani podrazumevani port isti kao u prethodnom zadatku. (1p)
- Predefinisati `getInputStream()` metod da vraća ulazni tok do odgovora od strane servera ukoliko je konekcija ostvarena, a `null` ako nije. (1p)
- Postarati se da je moguće bezbedno koristiti implementirani handler u višenitnom okruženju. (1p)
- Napisati jednostavan test - kreirati URL, otvoriti konekciju do resursa i ispisati sve podatke koje server pošalje. (2p)

Primer rada:

```
URL:   exchange://localhost
izlaz: Upit nije kompletan.

URL:   exchange://localhost:12345?iznos=5
izlaz: Upit nije kompletan.

URL:   exchange://localhost:7337?valuta=EUR&iznos=5
izlaz: 587.915

URL:   exchange//localhost?valuta=www&iznos=3
izlaz: Ne menjamo trazenu valutu

URL:   exchange//localhost?valuta=EUR&iznos=-1
izlaz: Iznos novca ne moze biti negativan broj
```

Računarske mreže, Ispit - JUN1 2022

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_mXGGXXX` 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!**

1. **Tokovi podataka i niti (15p)**
   Napisati program koji rekurzivno obilazi direktorijum i ispisuje broj linija u svim `.c` fajlovima koristeći niti.

   - Na Desktop-u se nalazi direktorijum `tests`. Obići pomenuti direktorijum i ispisati na standardni izlaz ukupan broj svih regularnih fajlova unutar tog direktorijuma. (3p)
   - Za svaki pronadjeni fajl sa ekstenzijom `.c` kreirati novi URL objekat koristeći `URL` klasu i `FILE` protokol. Ispisati kreirane URL-ove na standardni izlaz (videti primere ispod).

Putanja	Odgovarajući URL
`tests/dir/hello_world.c`	`FILE:///home/ispit/Desktop/tests/dir/hello_world.c`

     (2p)
   - Za svaki kreirani URL, kreirati zasebnu nit koja će otvoriti **baferisani** ulazni tok do resursa putem URL klase i pročitati sadržaj fajla (detalji obrade su u narednoj stavci). Kodnu stranu prilikom učitavanja postaviti na ASCII. (4p)
   - Na standardni izlaz ispisati ukupan broj linija u svim fajlovima iz prethodne stavke tako što će svaka nit prebrojati linije za fajl koji joj je dodeljen (videti primer ispisa ispod teksta zadatka). Pritom, paziti na sinhronizaciju niti ukoliko se koristi deljeni brojač. (5p)
   - Postarati se da program u slučajevima ispravno zatvori sve korišćene resurse. (1p)

```
ulaz:
izlaz: files:     10
       url:       FILE:///home/ispit/Desktop/tests/dir/hello_world.c
       url:       FILE:///home/ispit/Desktop/tests/dir/dir1/dir11/palind.c
       url:       FILE:///home/ispit/Desktop/tests/dir/dir1/smile.c
       url:       FILE:///home/ispit/Desktop/tests/dir/dir2/dir2q/pi.c
       result:    116
```

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. **Pogodi broj (TCP) (25p)**
Napraviti TCP klijent-server aplikaciju koja simulira igru "Pogodi broj". Na serveru se generiše ceo broj od 1 do 100, a zadatak klijenta je da taj broj pogodi,

- Implementirati klijentsku TCP aplikaciju koristeći `Java Socket API`. Klijent se povezuje na server na portu 12321, prima poruku "Pogodi koji broj od 1 do 100 sam zamislio", i šalje broj koji je odabrao. Nakon toga, prima poruke u formatu "Zamišljeni broj je manji/veći od toga". Klijent šalje novi broj, sve dok se broj ne pogodi, nakon čega dobija poruku "Čestitam! Pogodili ste broj." (7p)
- Implementirati serversku TCP aplikaciju koristeći *Java Socket API*. Uloga servera je da osluškuje na portu 12321, prihvata klijente i pokreće zasebnu nit za svakog klijenta. (4p)
- Pojedinačne niti koje obradjuju klijente imaju ulogu onog koji zamišlja broj. Potrebno je generisati jedan ceo broj od 1 do 100 i obavestiti klijenta da je igra počela porukom: "Pogodi koji broj od 1 do 100 sam zamislio". Nakon toga, prima broj koji je klijent poslao, proverava da li je veći ili manji od zamišljenog broja, i šalje odgovarajuću poruku. Igra se završava kada klijent pogodi broj, nakon cega dobija poruku da je broj pogodjen. (11p)
- Postarati se da su svi resursi ispravno zatvoreni u slučaju naglog isključivanja klijenta ili drugih izuzetaka. (3p)

Primer rada:

```
server : Pogodi koji broj od 1 do 100 sam zamislio
klijent: 50
server : Zamisljeni broj je veci od toga
klijent: 75
server : Zamisljeni broj je veci od toga
klijent: 87
server : Zamisljeni broj je manji od toga
klijent: 81
server : Zamisljeni broj je manji od toga
klijent: 78
server: Cestitam! Pogodili ste broj!
```

3. **UDP: Frankenštajnova rečenica (20p)**
Implementirati UDP klijent-server Java aplikaciju koja omogućava klijentima da konstruišu "Frankenštajnovu"rečenicu spajajući reči iz predefinisanog teksta.

- Napraviti metod unutar `UDPServer` klase koji kreira rečenicu spajajući reči iz fajla `frankenstajn.txt`. Argument metoda je kolekcija rednih brojeva reči u fajlu, u rastućem poretku. (3p)
- Napisati Java aplikaciju koja ima ulogu UDP klijenta koristeći *Java Datagram API*. Klijent sa standardnog ulaza učitava nisku, koja sadrži u rastućem poretku proizvoljan broj neoznačenih celih brojeva. Brojevi predstavljaju redne brojeve reči u fajlu sa serverske strane iz kojeg će se odabirati reči. Poslati datagram koji sadrži učitanu rečenicu serveru. Zatim primiti datagram koji predstavlja odgovor servera i rezultat ispisati na standardni izlaz. (5p)
- Napisati Java klasu koja ima ulogu lokalnog UDP servera koji osluškuje na portu 12345 koristeći *Java Datagram API*. Server prihvata datagrame od klijenata i za svaki prihvaćen datagram ispisuje na standardni izlaz `IP` adresu pošiljaoca i redni broj pristiglog datagrama. (5p)
- Za svaki prihvaćen datagram, server klijentu koji je poslao datagram šalje odgovor. Sadržaj odgovora je rečenica koju je potrebno kreirati nadovezivanjem reči čije redne brojeve u fajlu je poslao klijent, koristeći metod opisan u prvoj tački. Ignorisati nevalidne redne brojeve. (6p)
- Postarati se da su svi resursi ispravno zatvoreni u slučaju izuzetka. (1p)

Računarske mreže, Ispit - JUN2 2022

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_mXGGXXX` 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!**

1. **Tokovi podataka i niti (15p)**
   Napisati program koji obradjuje URL-ove unesene u pregledač i na standardni izlaz ispisuje odgovarajuće poruke.

   - U direktorijumu `tests` na Desktop-u se nalazi datoteka `log.txt`, koja predstavlja log unesenih URL-ova u nekom pregledaču, od kojih su neki validni, a neki ne. Svaku liniju iz log datoteke treba obraditi u zasebnoj niti na način naveden u nastavku. (2p)
   - Proveriti da li je URL validan. Ako nije, ispisati na standarni izlaz poruku u formatu: (2p)

     ```
     BADURL <LOS_URL_IZ_LOG_FAJLA>
     ```

   - U slučaju da je protokol `FILE`, proveriti da li ta datoteka postoji na našem sistemu. Ako postoji, otvoriti je putem URL klase i prebrojati linije u njoj, a poruku ispisati u formatu: (3p)

     ```
     OK <PUTANJA_DO_FAJLA> <BROJ_LINIJA>
     ```

   - U slučaju da datoteka ne postoji, ispisati poruku u formatu: (2p)

     ```
     NOTFOUND <PUTANJA_DO_FAJLA>
     ```

   - U slučaju bilo kog drugog protokola, "proslediti" zahtev dalje ispisom poruke u sledećem formatu: (3p)

     ```
     FORWARD <POCETNI_URL> [<TIMESTAMP>]
     ```

   - Voditi računa o tome da su resursi pravilno zatvoreni. (1p)
   - Sinhronizovati ispis na standardni izlaz. (2p)

```
primer:
ulaz:
  http://www.matf.bg.ac.rs/dir1/dir2/file1.txt
  sftp://2001:0db8:85a3:::8a2e:0370:7334/dir1/text.txt
  https://test.net/path/image.jpg
  FILE:///home/ispit/Desktop/tests/dir1/test1.txt
  file:///home/ispit/books/rm.pdf

izlaz:
  FORWARD http://www.matf.bg.ac.rs/dir1/dir2/file1.txt [Sat Jun 04 18:33:17 CEST 2022]
  BADURL sftp://2001:0db8:85a3:::8a2e:0370:7334/dir1/text.txt
  FORWARD https://test.net/path/image.jpg [Sat Jun 04 18:33:17 CEST 2022]
  OK /home/ispit/Desktop/tests/dir1/test1.txt 7
  NOTFOUND /home/ispit/books/rm.pdf
```

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. **Neblokirajući izbori (25p)**
U toku su izbori za studentski parlament. U fajlu `lista.txt` nalaze se kandidati za koje je moguće glasati. Napraviti klient-server Java aplikaciju koja vodi statistiku o izbornim glasovima.

   - Napisati Java klasu koja ima ulogu **blokirajućeg** TCP klijenta koristeći `Java Channels API`. Formirati konekciju sa lokalnim serverom na portu 12345. Nakon formiranja konekcije uneti glas sa standardnog ulaza i poslati ga serveru. Odgovor od servera ispisati na standardni izlaz. (8p)
   - Napisati Java klasu koja ima ulogu lokalnog **neblokirajućeg** TCP servera koji osluškuje na portu 12345 koristeći Java Channels API. Server vodi evidenciju o trenutnom broju glasova. Na svakih 5 sekundi, ako je do tada bar neko glasao, ispisati trenutni procenat glasova za svakog kandidata i trenutni procenat nevalidnih glasova. Validni su glasovi samo za kandidate iz fajla `lista.txt`. Za svaki primljeni glas od klijenta poslati odgovor. Ako je klijent poslao nevažeći glas, kao odgovor poslati obaveštenje da glas nije validan. U suprotnom poslati zahvalnicu na glasanju i trenutni procenat glasova za izabranog kandidata zaokružen na dve decimale. (15p)
   - Postarati se da su svi resursi ispravno zatvoreni u slučaju izuzetka (2p)

   Primer rada:

```
ulaz : Marko
izlaz: Hvala sto ste glasali, trenutni procenat glasova za vaseg kandidata je 1.00

ulaz : Pera
izlaz: Vas glas nije validan

ulaz : maja
izlaz: Hvala sto ste glasali, trenutni procenat glasova za vaseg kandidata je 0.33
```

3. **Protocol handlers (15p)**
Implementirati podršku za URL-ove koji koriste `evote` protokol. Opis protokola je dat 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)
   - Omogućiti slanje upita pomoću parametra `kandidat` URL-a, primer (paziti na enkodiranje karaktera unutar URL-a):

     ```
     evote://localhost:1337?kandidat=Marko+Markovic
     ```

     Server šalje nazad rezultat koji klijent ispisuje kao u primeru ispod. (5p)
   - Ukoliko port nije naveden unutar URL-a, iskoristiti predefinisani podrazumevani port isti kao u prethodnom zadatku. (1p)
   - Predefinisati `getInputStream()` metod da vraća ulazni tok do odgovora od strane servera ukoliko je konekcija ostvarena, a `null` ako nije. (1p)
   - Postarati se da je moguće bezbedno koristiti implementirani handler u višenitnom okruženju. (1p)
   - Napisati jednostavan test - kreirati URL, otvoriti konekciju do resursa i ispisati sve podatke koje server pošalje. (2p)

   Primer rada:

```
URL:   evote://localhost
izlaz: Upit nije kompletan.

URL:   evote://localhost:12345?test=5
izlaz: Upit nije kompletan.

URL:   evote://localhost:7337?kandidat=Marko
izlaz: Hvala sto ste glasali, trenutni procenat glasova za vaseg kandidata je 1.00

URL:   evote//localhost?kandidat=Pera
izlaz: Vas glas nije validan
```

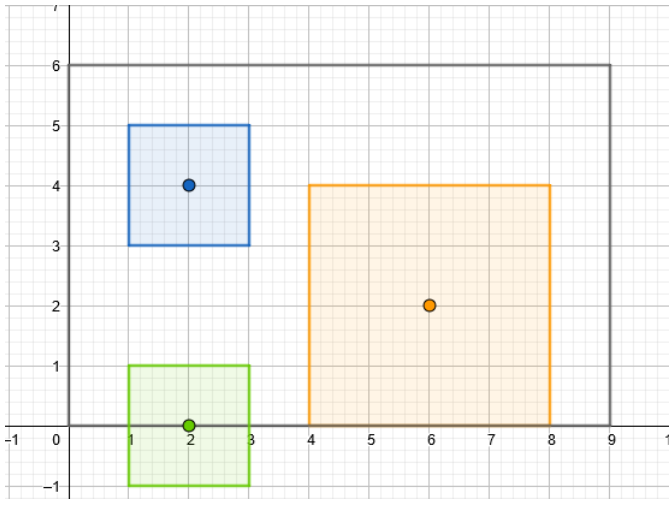
Računarske mreže, Ispit - SEP1 2022

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_mXGGXXX` 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!**

1. **Magične matrice (20p)**
   Napisati Java aplikaciju koja proverava da li su zadate matrice *magične*. Kvadratna matrica je magična ako su sume elemenata u svim vrstama, svim kolonama i na glavnoj i sporednoj dijagonali jednake. U direktorijumu `tests` na `Desktop-u` se nalaze datoteke koje sadrže matrice — prvi broj predstavlja broj vrsta kvadratne matrice, nakon čega slede elementi matrice. Pretpostaviti da su dimenzije matrice ispravne i da su svi elementi matrice različiti.

   - Rekurzivno obići direktorijum `tests` i za svaku datoteku pokrenuti zasebnu nit koja je pretražuje. (4p)
   - Za svaku datoteku ispisati njen naziv i da li je zadata matrica magična ili ne. (6p)
   - Postarati se da se ispisi svake niti na standardni izlaz ne prepliću. (5p)
   - Ispisati koliko je ukupno pronađenih magičnih matrica (paziti na sinhronizaciju niti ukoliko se koristi deljeni brojač!). (3p)
   - Postarati se da program ispravno obrađuje specijalne slučajeve i ispravno zatvara sve korišćene resurse. (2p)

2. **Non-Blocking IO (25p)**
   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. (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 svakom od skenera šalje broj drugih skenera cija se pokrivena teritorija preseca sa pozicijom trenutnog skenera. (9p)
- Server salje indikator svim skenerima ukoliko procenat pokrivenosti terena dostigne 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)

3. **UDP Sockets (15p)**

Napisati Java aplikaciju koja računa zbir svih prirodnih brojeva u zadatom opsegu.

- Napisati Java klasu koja ima ulogu UDP klijenta. Sa standardnog ulaza učitati dva prirodna broja. Poslati UDP datagram koji sadrži dva učitana broja lokalnom serveru na portu 12345 koristeći `DatagramPacket` klasu. (3p)
- Napisati Java klasu koja ima ulogu UDP servera. Slušati na portu 12345 i primati datagrame od klijenata koristeći `DatagramPacket` klasu. Sadržaj svakog datagrama su dva prirodna broja. (3p)
- Server očitava sadržaj datagrama i računa zbir svih brojeva u opsegu [N1, N2] (inkluzivno). (3p)
- Postarati se da je operacija računanja zbira složenosti O(1). (2p)
- U slučaju nevalidnog opsega (negativni brojevi ili prvi broj veći od drugog), poslati klijentu poruku "Nevalidan opseg!". (2p)
- Postarati se da su svi resursi ispravno zatvoreni u slučaju izuzetka. (2p)

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!

Računarske mreže, Ispit - SEP2 2022

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_mXGGXXX` 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!**

1. **Matrično množenje (15p)**
   Napraviti Java aplikaciju koja koristeći niti množi dve kvadratne matrice.

   - Kao ulaz u program se daje putanja do tekstualnih fajlova u kojima se nalaze kvadratne matrice koje je potrebno pomnožiti - po jedna u svakom fajlu. Učitati i ispisati matrice na standardni izlaz. (3p)
   - Ukoliko matrice ne mogu da se množe, ispisati poruku i prekinuti izvršavanje programa. (1p)
   - Označimo dimenzije matrica sa $n \times n$. Pokrenuti n^2 niti i postarati se da svaka nit računa jedan element rezultujuće matrice. Ispisati rezultujuću matricu na standardni izlaz. (5p)
   - Računati zbir elemenata rezultujuće matrice tokom rada svake niti. Kada svaka nit izračuna svoju vrednost, ažurira globalni zbir. Postarati se da nema trke za podacima - obezbediti kritičnu sekciju proizvoljnim mehanizmom. (4p)
   - Postarati se da program ispravno obrađuje specijalne slučajeve (npr. ako fajl ne postoji na datoj putanji) i ispravno zatvoriti sve korišćene resurse u slučaju izuzetka. (2p)

2. **NBIO (25p)**
   Bliži se kraj ispitnog roka, napravi klient-server Java aplikaciju koristeći TCP Sockets/Channels API koja ce pomoći studentima da izračunaju koliko stranica dnevno moraju da nauče kako bi uspešno položili ispite.

   - Napisati Java klasu koja ima ulogu **blokirajućeg** TCP klijenta koristeći `Java Channels API`. Klijent formira konekciju sa lokalnim serverom na portu `12345`. Nakon formiranja konekcije, klijent šalje serveru zahteve, red po red, sve do unosa niske `"exit"`. Zahtev čine, razdvojeni razmakom, datum od kog student planira da počne da uči, datum održavanja ispita, broj strana koje student treba da nauči do tada (datumi se unose u formatu `dan/mesec/godina`). Ukoliko je datum početka izostavljen postavlja se na današnji datum. Pre učitavanja narednog reda potrebno je sačekati odgovor servera i ispisati ga na standardni izlaz. Odgovor servera predstavlja broj strana koje student treba dnevno da nauči kako bi položio taj ispit. (8p)
   - Napisati Java klasu koja ima ulogu **neblokirajućeg** TCP servera koji osluškuje na portu `12345` koristeći `Java Channels API`. Server prima zahteve od klijenata i nakon svakog primljenog zahteva vraća odgovor. Ukoliko je broj preostalih dana do ispita manji od 7 dodatno obavestiti studenta o tome. Ukoliko su uneti datumi takvi da je datum početka veći ili jednak od datuma održavanja ispita obavestiti studenta da neće položiti ispit u ovom roku (13p)
   - Postarati se da svi resursi budu pravilno zatvoreni. (4p)

   Primer rada:

   ```
   ulaz: 12/09/2022 23/09/2022 220      izlaz: 20

   ulaz: 20/09/2022 23/09/2022 636      izlaz: 212, ostalo je manje od 7 dana

   ulaz: 10/09/2022 200                 izlaz: Vidimo se u narednom roku.
   ```

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. **UDP (15p)**
Napisati Java aplikaciju koja dohvata informacije o studentu na osnovu njegovog indeksa.

- Napisati Java klasu koja ima ulogu UDP klijenta. Poslati inicijalni datagram koji sadrži indeks u formatu `XXX/GGGG` i jednu od komandi `prosek` ili `ime` lokalnom serveru koji osluškuje na portu 12321. Primiti datagram od servera koji sadži informaciju koja je zatražena. Podatke u datagramu kodirati proizvoljno. (4p)
- Napisati Java klasu koja ima ulogu servera koji osluškuje na portu 12321.
  - Iz tekstualnog fajla `studenti.txt` učitati informacije o studentima i keširati ih na serveru. Moguće je koristiti proizvoljnu kolekciju podataka. (2p)
  - Primiti datagram od klijenta, isparsirati komandu, i vratiti datagram sa zatraženom informacijom o studentu. (4p)
  - U slučaju nepostojećeg indeksa, ili nevalidne komande, vratiti klijentu poruku "Zahtev nije validan". (2p)
- Postarati se da su svi resursi pravilno zatvoreni u slučaju izuzetka. (3p)

Primer izvršavanja:

```
klijent: 002/2018 ime                    server: Pera Peric

klijent: 100/2018 prosek                 server: 6.3

klijent: 101/2018 prosek                 server: Zahtev nije validan!

klijent: 100/2018 email                  server: Zahtev nije validan!
```