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)