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
```