Naziv: višedretvena igra "Potapljanje brodova" putem socket-a

Naziv projekta: **{LDAP_korisničko_ime}_zadaca_1**

Sve nove klase trebaju biti u paketu **org.foi.nwtis.{LDAP_korisničko_ime}.zadaca_1**. Za rad s postavkama treba koristiti Java biblioteku iz vjezba_03_2. Vlastite biblioteke treba smjestiti na direktorij .\lib unutar projekta. Klase i metode trebaju biti komentirane u javadoc formatu. Prije predavanja projekta potrebno je napraviti Clean na projektu. Zatim cijeli projekt (korijenski direktorij treba biti**{LDAP_korisničko_ime}_zadaca_1)** sažeti u .zip (NE .rar) format s nazivom **{LDAP_korisničko_ime}_zadaca_1.zip** i predati u Moodle. Uključiti izvorni kod, primjere datoteka konfiguracijskih podataka (.txt i .xml) i popunjeni obrazac za zadaću pod nazivom**{LDAP_korisničko_ime}_zadaca_1.[doc | pdf]** (u korijenskom direktoriju projekta).

Plavila igre:

Server generira početno stanje ploče na temelju postavki za veličinu ploče, broja igrača i broja brodova po igraču. Korisnici se moraju prvo prijaviti za igru sve dok ima slobodnog mjesta. Samo prijavljeni korisnici - igrači mogu igrati igru. Igrači mogu igrati svoj potez bez obrzira na redoslijed prijavljivanja u igru, s time da jedan igrač ne može imati 2 poteza više od bilo kojeg igrača. Ako pokuša igrati, to mu neće biti dozvoljeno a dobit će informaciju na koliko igrača treba čekati. Igrač kojem su potopljeni svi brodovi ne može igrati nove poteze. Pobjednik je onaj igrač koji je potopio zadnji brod jedinog preostalog igrača suparnika.

Nakon svakog poteza igrača na konzoli servera ispisuje se trenutno stanje na ploči (pozicije brodova igrača i pozicije potopljenih brodova) tako da se vidi koji brodovi pripadaju pojedinom igraču i koji brodovi su potopljeni od pojedinog igrača.

U zadaći **ne smiju se koristiti** klase Timer, TimerTask i druge izvedene klase za upravljanje dretvama i sl!

Program se sastoji od 4 dijela od kojih se samo jedan može odabrati kod pokretanja programa:

1.server (opcija -server) - prima zahtjeve ostala tri dijela

2.administrator (opcija -admin)

3.klijent (opcija -user)

4.prikaz (opcija -show).

Program se može izvršavati više puta kako bi se koristile različiti dijelovi. Izvršavanje programa započinje utvrđivanjem vrste rada (server, admin, user, show) na bazi opcija u parametarima. Komunikacija pojedinih vrsta korisnika (administrator, klijent, prikaz) i servera temelji se na jednostavnom protokolu koji ima određenu sintaksu, a sadrži skup komandi. Komande i njihova sintaksa objašnjeni su kod pojedine vrsta korisnika. Logično je da se prvo pokreće server, a nakon njega korisnici. Komunikaciju otvara jedna od vrsta korisnika tako da šalje zahtjev serveru u obliku određene komande. Server provodi analizu zahtjeva i ako je zahtjev u redu provodi određenu operaciju te vraća status operacije i ostale potrebne podatke. Kada zahtjev nije u redu vraća primjereni status operacije. Kontrola parametara treba se obaviti u posebnim funkcijama unutar klasa koje su zadužene za pojedine dijelove programa. Treba pripremiti JUnit testove za metode za

kontrolu parametara. Poželjni su i JUnit testovi drugih metoda.

U radu programa koristi se datoteka s postavkama koja mora imali ekstenziju txt ili xml a može biti lokalna, s apsolutnim ili relativnim nazivom ili na poslužitelju s internetskom adresom (npr. NWTiS_dkermek_zadaca_1.txt, E:\dkermek_zadaca_1.xml, http:\\localhost\zadaca_1.xml). Datoteka može sadržavati sljedeće elemente:

- •port - port na kojem radi server, između 8000 i 9999
- •evidDatoteka - datoteka u koju se provodi serijalizacija evidencije rada (važeći podaci o igračima, pozicijama i potezima igrača). Datoteka je lokalna, s apsolutnim ili relativnim nazivom (npr. evidencija.bin, d:\NWTiS\evidencija.bin,...)
- •intervalSerijalizacije - broj sekundi za interval serijalizacija podataka
- •adminKorIme - korisničko ime administratora sustava
- •adminKorLozinka - lozinka administratora sustava.
- •brojDretvi - maksimalan broj dretvi kod servera koje mogu opsluživati korisnike
- •razmakDretvi - maksimalan broj sekundi za razmak kod pokretanja dretve - nije potrebno - IZBAČENO
- •intervalDretve - broj sekundi za interval rada dretve - nije potrebno - IZBAČENO
- •brojX - broj polja ploče u X osi (3 do 10)
- •brojY - broj polja ploče u Y osi (3 do 10)
- •brojIgraca - broj igrača za igru (2 do 5)
- •brojBrodova - broj brodova po igraču (2 do 5)

Program radi kao server ako zadovoljava sljedeći oblik parametara:

-server -konf datoteka(.txt | .xml) [-load]

Server na početku provjerava postoji li datoteka konfiguracije (opcija -konf) te prekida rad ako ne postoji. Ako postoji, učitava postavke iz datoteke konfiguracije. Ako je upisana opcija -load, server provjerava postoji li datoteka sa serijaliziranim podacima (postavka evidDatoteka) te ju učitava ako postoji. Ako ne postoji, prvo generira ploču dimenzije X * Y polja (postavka brojX, postavka brojY). Zatim za svakog igrača (1 - postavlja brojIgraca) generira i pamti pozicije njegovih brodova. Generator slučajnih brojeva u intervalu 0.0 do 1.0 na bazi funkcije Math.random()
(adresa http://docs.oracle.com/javase/8/docs/api/java/lang/Math.html#random()) .Dva broda (neovisno o igraču) ne mogu biti na istoj poziciji. Server ima ima ulogu socket servera na određenom portu (postavka port) te može imati maksimalnu veličinu reda čekanja (postavka brojDretvi). Upoznati se s klasom ServerSocket (http://docs.oracle.com/javase/8/docs/api/java/net/ServerSocket.html) i klasom Socket (http://docs.oracle.com/javase/8/docs/api/java/net/Socket.html).

U nastavku server kreira objekt grupe dretvi (naziv {LDAP_korisničko_ime}) za posluživanje klijenata, a unutar njega kreira potreban broj dretvi (postavka brojDretvi) pri čemu im je naziv jednak {LDAP_korisničko_ime}_redniBroj te ih smjesti u izabranu strukturu (npr. niz). Server čeka na uspostavljanje veze na socketu od korisnika i kada se veza uspostavi on tu vezu predaje sljedećeoj slobodnoj dretvi, s time da dretve u kružnom

redoslijedu opslužuju zahtjeve korisnika kako bi se ostvarilo približno jednako zapošljavanje svih dretvi. Prilikom traženja slobodne dretve treba ispisivati na ekran koje dretve su zauzete te koja je na kraju izabrana kao slobodna. Ako nema slobodne dretve tada se korisniku vraća odgovor ERROR 80; tekst (tekst objašnjava razlog pogreške). Nakon toga server ponovno čeka na uspostavljanje veze i postupak se nastavlja. Dretva nakon što obradi pridruženi zahtjev korisnika postaje slobodna pa joj se može pridružiti novi zahtjev kada na nju dođe red.

Dretva iz dobivene veze na socketu preuzima tokove za ulazne i izlazne podatke prema korisniku. Na temelju ulaznih podataka provodi se analiza zahtjeva korisnika. Dozvoljene komande opisane su u posebnom dijelu. Ako sintaksa nije ispravna ili komanda nije dozvoljena tada se korisniku vraća odgovor ERROR 90; tekst (tekst objašnjava razlog pogreške).

Kada dretva primi zahtjev od korisnika ona ažurira podatke u evidenciji rada. Postupak ažuriranja mora se provoditi međusobno isključivo u odnosu na druge dretve koje upisuju podatke u evidenciju. Evidencija rada treba prikupljati podatke o važećim podacima o igračima, pozicijama i potezima igrača. Za podatke treba koristiti vlastitu klasu koja se može serijalizirati. Server treba provoditi serijalizaciju podataka evidencije putem dretve koja provodi serijalizaciju u pravilnim vremenskim intervalima (postavka intervalSerijalizacije). Postupak serijalizacije mora se provoditi međusobno isključivo u odnosu na druge dretve koje upisuju podatke u evidenciju. Kod serijalizacije podataka postavka evidDatoteka nosi kostur za dodjelu naziva datoteke tako da se nakon prvog dijela naziva, a prije ekstenzije (ako postoji) doda redni broj po formatu **9999**. Za formatiranje:

- •datumskih podataka preporučuje se klasa **java.text.SimpleDateFormat** (http://docs.oracle.com/javase/8/docs/api/java/text/SimpleDateFormat.html)
- •brojčanih podataka preporučuje se klasa **java.text.DecimalFormat** (http://docs.oracle.com/javase/8/docs/api/java/text/DecimalFormat.html).

Program radi kao administrator servera ako zadovoljava sljedeći oblik parametara:

-admin -s [ipadresa | adresa] -port port -u korisnik -p lozinka [-pause | -start | -stop | -stat | -new]

Dozvoljene vrijednost za opcije:

- •ipadresa je adresa IPv4 (npr. 127.0.0.1, 192.168.15.1)
- •adresa je opisni naziv poslužitelja (npr. localhost, dkermek.nwtis.foi.hr)
- •port može biti u intervalu između 8000 i 9999.
- •korisnik može sadržavati mala i velika slova, brojeve i znakove: _, -
- •lozinka može sadržavati mala i velika slova, brojeve i znakove: _, -, #, !

Administrator servera šalje komandu serveru putem socketa na temelju upisanih parametara i traži izvršavanja određene akcije:

- •USER korisnik; PASSWD lozinka; PAUSE;

•-upisan parametar *-pause* pa provjerava se postoji li korisnik (postavka adminKorIme) i njemu pridružena lozinka (postavka adminKorLozinka). Ako je u redu i server nije u stanju pause, privremeno prekida prijem svih komandi osim administratorskih. Korisniku se vraća odgovor OK. Kada nije u redu, korisnik nije administrator ili lozinka ne odgovara, vraća se odgovor ERROR 00; tekst (tekst objašnjava razlog pogreške). Ako je u stanju pause vraća se odgovor ERROR 01; tekst (tekst objašnjava razlog pogreške).

•USER korisnik; PASSWD lozinka; START;

•-upisan parametar *-start* pa provjerava se postoji li korisnik (postavka adminKorIme) i njemu pridružena lozinka (postavka adminKorLozinka). Ako je u redu i server je u stanju pause, nastavlja prijem svih komandi. Korisnku se vraća odgovor OK. Kada nije u redu, korisnik nije administrator ili lozinka ne odgovara, vraća se odgovor ERROR 00; tekst (tekst objašnjava razlog pogreške). Ako nije u stanju pause vraća se odgovor ERROR 02; tekst (tekst objašnjava razlog pogreške).

•USER korisnik; PASSWD lozinka; STOP;

•-upisan parametar *-stop* pa provjerava se postoji li korisnik (postavka adminKorIme) i njemu pridružena lozinka (postavka adminKorLozinka). Ako je u redu prekida prijem komandi, serijalizira svoju evidenciju i završava rad. Korisniku se vraća odgovor OK. Kada nije u redu, korisnik nije administrator ili lozinka ne odgovara, vraća se odgovor ERROR 00; tekst (tekst objašnjava razlog pogreške). Ako nešto nije u redu s prekidom rada ili serijalizacijom vraća se odgovor ERROR 03; tekst (tekst objašnjava razlog pogreške).

•USER korisnik; PASSWD lozinka; STAT;

•-upisan parametar *-stat* pa provjerava se postoji li korisnik (postavka adminKorIme) i njemu pridružena lozinka (postavka adminKorLozinka). Ako je u redu korisniku se vraća odgovor OK; <CRLF> i zatim na bazi evidencije vraća podatke o stanju na ploči (pozicije brodova igrača i pozicije potopljenih brodova) tako da se vidi koji brodovi pripadaju pojedinom igraču i koji brodovi su potopljeni od pojedinog igrača). Kada nije u redu, korisnik nije administrator ili lozinka ne odgovara, vraća se odgovor ERROR 00; tekst (tekst objašnjava razlog pogreške). Ako nešto nije u redu s evidencijom vraća se odgovor ERROR 05; tekst (tekst objašnjava razlog pogreške).

•USER korisnik; PASSWD lozinka; NEW;

•-upisan parametar *-new* pa provjerava se postoji li korisnik (postavka adminKorIme) i njemu pridružena lozinka (postavka adminKorLozinka). Ako je u redu prekida prijem komandi, serijalizira svoju evidenciju i generira novu igru. Korisniku se vraća odgovor OK. Kada nije u redu, korisnik nije administrator ili lozinka ne odgovara, vraća se odgovor ERROR 00; tekst (tekst objašnjava razlog pogreške). Ako nešto nije u redu s prekidom rada ili serijalizacijom vraća se odgovor ERROR 03; tekst (tekst objašnjava razlog pogreške).

Program radi kao klijent servera ako zadovoljava sljedeći oblik parametara:

-user -s [ipadresa | adresa] -port port -u korisnik [-x {x koordinata} -y {y koordinata} | -stat]

Dozvoljene vrijednost za opcije:

- •ipadresa je adresa IPv4 (npr. 127.0.0.1, 192.168.15.1)
- •adresa je opisni naziv poslužitelja (npr. localhost, dkermek.nwtis.foi.hr)
- •port može biti u intervalu između 8000 i 9999.
- •korisnik može sadržavati mala i velika slova, brojeve i znakove: _, -
- •x može biti u intervalu između 1 i brojX
- •y može biti u intervalu između 1 i brojY

Klijent servera spaja se na server i nakon toga šalje komandu serveru putem socketa i traži izvršavanja određene akcije:

- •USER korisnik; PLAY; Korisnik traži da bude igrač. Ako ima slobodnog mjesta, pamti se korisnik kao igrač i vraća mu se odgovor OK; <CRLF> slijede podaci o veličini ploče, broju igrača i brodova po igraču u obliku brojX, brojY, brojIgraca, brojBrodova <CRLF> i zatim koordinate njegovih brodova po formatu [x,y]{2 do postavlka brojBrodova}. Korisnik može za jednu igru samo jednom obaviti prijavu. Ako nije u redu, vraća se odgovor ERROR 10; tekst (tekst objašnjava razlog pogreške)
- •

  USER korisnik; [x,y]; Ako je prijavljen igrač vraća mu se odgovor OK; i zatim slijedi {0 | 1 | 2 (n) | 3 | 9} pri čemu 0 znači da nije pogodio brod, 1 da je pogodio broj (može se pogoditi i vlastiti brod, ili brod koji je već potopljen), 2 da mora čekati dok drugi igrači ne odigraju svoj potez a n označava koliko igrača se čeka, 3 da su mu potopljeni svi brodovi i nema prvo dalje igrati igru, 9 da je pobjednik igre. Ako nije u redu, vraća se odgovor ERROR 10; tekst (tekst objašnjava razlog pogreške).
- •

  •USER korisnik; STAT; Ako je prijavljen igrač i igra je završila vraća mu se odgovor OK; <CRLF> i zatim na bazi evidencije vraća podatke o stanju na ploči (pozicije brodova igrača i pozicije potopljenih brodova) tako da se vidi koji brodovi pripadaju pojedinom igraču i koji brodovi su potopljeni od pojedinog igrača.. Ako nije u redu, vraća se odgovor ERROR 10; tekst (tekst objašnjava razlog pogreške).

Program radi kao formatirani prikaz serijaliziranih podataka evidencije servera ako zadovoljava sljedeći oblik parametara:

-show -s datoteka

Dozvoljene vrijednost za opcije:

- •datoteka u koju je provedena serijalizacija podataka evidencije o radu, može biti lokalna, s apsolutnim ili relativnim nazivom, ili na poslužitelju s internetskom adresom (npr. evidencija.bin, d:\NWTiS\radKorisnika.bin, http:\\localhost\evidencija.bin).

Prikaz podataka treba biti na takav način da se vidi početno stanje na ploči, svaki pojedinačni potez igrača, novo stanje na ploči i ako je gotova igra, koji igrač je pobjednik.

Za pretvaranje serjializiraniih podataka iz evidencije u čitljiv i formatirani oblik. Za formatiranje:

- •datumskih podataka preporučuje se klasa **java.text.SimpleDateFormat** (http://docs.oracle.com/javase/8/docs/api/java/text/SimpleDateFormat.html)
- •brojčanih podataka preporučuje se klasa **java.text.DecimalFormat** (http://docs.oracle.com/javase/8/docs/api/java/text/DecimalFormat.html).

Zadaća_1: višedretveni sustav upravljanja putem socket-a

1.U NetBeans kreiranje projekta **{LDAP_korisničko_ime}_zadaca_1** kao Java aplikacije, na direktorij **{LDAP_korisničko_ime},** bez kreiranja glavne klase i kao glavni projekt

2.Kreiranje paketa **org.foi.nwtis.{LDAP_korisnik}.zadaca_1**

3.Kreirati direktorij .\lib u korijenskom direktoriju projekta

4.Kopirati Java biblioteku iz prethodnog zadatka (vjezba_03_2.jar) u direktorij .\lib

5.Dodavanje Java biblioteke iz prethodnog zadatka: desna tipka na mišu na Libraries / Add Jar/Forder / Odabrati .\lib\vjezba_03_2.jar

6.Kreiranje klase Zadaca_{LDAP_korisnik}_1 u paketu org.foi.nwtis.{LDAP_korisnik}.zadaca_1. Klasa provjerava korektnost prve upisane opcije te služi za usmjeravanje na jedan od četiri načina rada programa, odnosno kreira se objekt pojedine klase (u nastavku) i prepušta mu se izvršavanje. Preporučuje se koristiti dopuštene izraze (RegEx). Primjer TestOpcija.java nalazi se nakon opisa vježbe/zadaće.

- •Java RegEx tutorial http://docs.oracle.com/javase/tutorial/essential/regex/
- •Korisni primjeri za RegEx http://www.mkyong.com/regular-expressions/10-java-regular-expression-examples-you-should-know/

7.Kreiranje klase **ServerSustava**. Prvo se provjeravaju upisane opcije, preporučuje se koristiti dopuštene izraze. Učitavaju se postavke iz datoteke. Kreira se i pokreće dretva za serijalizaciju evidencije (klasa SerijalizatorEvidencije). Kreira se grupa dretvi i za nju potreban broj dretvi (klasa ObradaZahtjeva). Otvara se ServerSocket (slično primjerima ClientTester.java i TinyHttpd.java s 4. predavanja) na izabranom portu i čeka zahtjev korisnika u beskonačnoj petlji. Kada se korisnik spoji na otvorenu vezu ona se predaje slobodnom objektu dretve (klasa ObradaZahtjeva) i pokreće se njezino izvršavanje. Dretve opslužuju zahtjeve korisnika u kružnom redoslijedu. Ako nema slobodne dretve tada se korisniku vraća odgovor ERROR 80; tekst (tekst objašnjava razlog pogreške). Nakon toga server ponovno čeka na uspostavljanje veze i postupak se nastavlja. Dretva nakon što obradi pridruženi zahtjev korisnika postaje slobodna pa joj se može pridružiti novi zahtjev kada na nju dođe red.

8.Kreiranje klase **ObradaZahtjeva.** Kreiranje konstruktora klase i metode za prijenos potrebnih podataka. Dretva iz dobivene veze na socketu preuzima tokove za ulazne i izlazne podatke prema korisniku. Dretva preuzima podatke koje šalje korisnik putem ulaznog toka podataka, provjerava korektnost komandi iz zahtjeva. Preporučuje se koristiti

dopuštene izraze. Za prvo testiranje servera može se koristiti primjer s 4. predavanja Primjer33_3.java. Na kraju dretva šaljem podatke korisniku putem izlaznog toka podataka. Za pojedine zahtjeve radi se s datotekom konfiguracije (čitanje te prijenos korisniku, čitanje od korisnika te kreiranje). Za svaku vrstu komande kreira se posebna metoda koja odrađuje njenu funkcionalnost.

9.Kreiranje klase **Evidencija**. Služi za evidenciju podataka i može se serijalizirati.Treba odrediti dodatne klase i varijable u koje će se pridružiti vrijednosti. Potrebno je voditi brigu o međusobnom isključivanju dretvi kod pristupa evidenciji rada i sl.

10.Kreiranje klase **SerijalizatorEvidencije**. Kreira se konstruktor klase u koji se prenose podaci konfiguracije. Služi za serijalizaciju podataka. Izvršava serijalizaciju evidencije u pravilnim vremenskim ciklusima. Potrebno je voditi brigu o međusobnom isključivanju dretvi kod pristupa evidenciji rada i sl.

11.Kreiranje klase **KlijentSustava**. Prvo se provjeravaju upisane opcije, preporučuje se koristiti dopuštene izraze. Spaja na server i nakon toga šalje komandu serveru putem socketa i traži izvršavanja određene akcije. Primljeni odgovor se ispisuju na ekranu korisnika.

12.Kreiranje klase **AdministratorSustava**. Prvo se provjeravaju upisane opcije, preporučuje se koristiti dopuštene izraze. Učitavaju se postavke iz datoteke. Objekt klase spaja se na server i šalje komandu(e) u zahtjevu. Primljeni odgovori se ispisuju na ekranu korisnika. Za svaku vrstu opcija kreira se posebna metoda koja odrađuje njenu funkcionalnost.

13.Kreiranje klase **PregledSustava**. Prvo se provjeravaju upisane opcije, preporučuje se koristiti dopuštene izraze. Otvara i čita datoteku sa serijaliziranim podacima evidencije i ispisuje ih u prikladnom/čitljivom i formatiranom obliku na ekran/standardni izlaz korisnika.

14.Kod prvog pokretanja programa s osobinom servera javlja se Sigurnosna stijena (Firewall) s pitanjem o blokiranju ili dozvoli rada programa. Potrebno je dozvoliti rad programu (Java SE...). Drugi način je da to uradimo unaprijed putem postavki veze koju koristimo (LAN. wireless) za Advanced, Settings, u kojima dodamo port (Add Port) koji će biti otvoren (slika).