Predmet: Napredne Web tehnologije i servisi
Ak. god.: 2019./2020.

Vježba: 4 – Zadaća 1. Višedretveni sustav "Informacije o avionima" putem mrežne utičnice/socket-a

Naziv projekta: **{LDAP_korisničko_ime}_zadaca_1**
Korijenski direktorij treba biti **{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. **Projekt se isključivo treba predati u formatu NetBeans projekta.** Prije predavanja projekta potrebno je napraviti Clean na projektu. Zatim cijeli projekt sažeti u **.zip** (NE .rar i sl.) format s nazivom **{LDAP_korisničko_ime}_zadaca_1.zip** i predati u Moodle. Uključiti izvorni kod, primjere datoteka konfiguracijskih podataka (.txt, .xml, .bin, .json) i popunjeni obrazac za zadaću pod nazivom **{LDAP_korisničko_ime}_zadaca_1.[doc | pdf]** (u korijenskom direktoriju projekta). Primjeri datoteka konfiguracijskih podataka (.txt, .xml, .bin, .json) trebaju sadržavati identične podatke. U radu programa datoteka konfiguracijskih podataka je smještena na direktoriju s kojeg se pokreće program (npr. NWTiS_dkermek_zadaca_1.txt). Program se može pokretati s različitih direktorija kako bi se mogao izvršavati s različitim datotekama konfiguracijskih podataka. Datoteka konfiguracijskih podataka može sadržavati sljedeće elemente:

Ključ	Vrijednost
port.simulator	Port na kojem radi server za simuliranje leta, između 8000 i 8999.
port.avioni	Port na kojem radi server za avione, između 9000 i 9999.
maks.cekaca	Maksimalni broj korisničkih zahtjeva koji čekaju na ostvarivanje veze kod servera, između 1 i 99.
datoteka.avioni	Datoteka u koju se provodi serijalizacija evidentiranih aviona. Datoteka je lokalna i smještena na direktorij s kojeg se pokreće porgram (npr. avioni.bin)
datoteka.aerodromi	Datoteka u kojoj se nalazi popis aerodroma sa podacima o lokaciji aerodroma.
datoteka.korisnika	Datoteka u kojoj se nalazi popis korisničkih imena i njihovih lozinki u XML formatu. Minimalno treba upisati podatke za 7 korisnika, od kojih je jedan „admin” s lozinkom „foi”.
interval.pohrane.aviona	Broj sekundi za svaki pravilan ciklus serijalizacije podatka o avionima, između 1 i 999.

Metode u klasama NE smiju imati više od 25 linija programskog koda, u što se ne broji definiranje metode, njenih argumenata i lokalnih varijabli. U jednoj liniji može biti jedna instrukcija. Linija ne može ima više od 132 znaka.

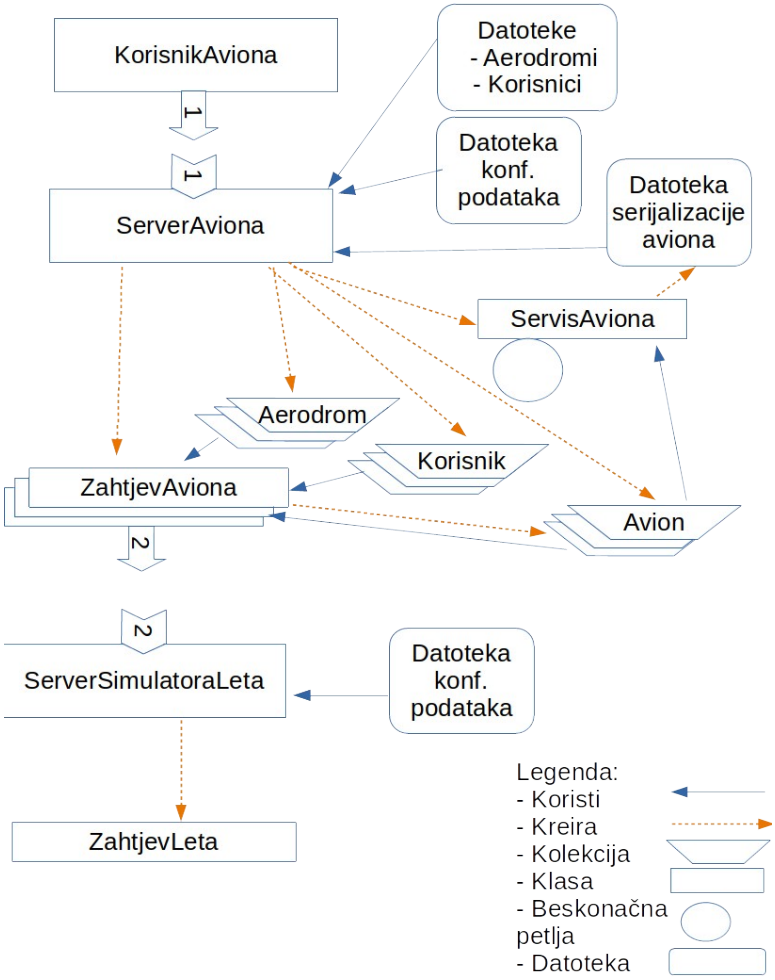
Treba pripremiti JUnit testove za sve metoda kod serverskog dijela s različitih tvrdnjama. Metode serverskog dijela NE SMIJU niti privatne.

Metode koje su nadjačane i jednostavne metode (get, set) nije potrebno komentirati u JavaDoc.

Boduju se dijelovi koji su rađeni nakon vježbi!

Opis rada sustava:

Sustav ima dva odvojena dijela: poslužiteljski/serverski i korisnički. Shema sustava prikazana je na slici.



Upoznati se s klasama:

https://docs.oracle.com/en/java/javase/13/docs/api/java.base/java/lang/Thread.html
https://docs.oracle.com/en/java/javase/13/docs/api/java.base/java/lang/ThreadGroup.html
https://docs.oracle.com/en/java/javase/13/docs/api/java.base/java/text/SimpleDateFormat.html
https://docs.oracle.com/en/java/javase/13/docs/api/java.base/java/text/DecimalFormat.html
https://docs.oracle.com/en/java/javase/13/docs/api/java.base/java/net/ServerSocket.html
https://docs.oracle.com/en/java/javase/13/docs/api/java.base/java/net/Socket.html
https://docs.oracle.com/en/java/javase/13/docs/api/java.base/java/util/Stack.html
https://docs.oracle.com/en/java/javase/13/docs/api/java.base/java/util/ArrayList.html

Pokretavanje **serverskog programa** za simulaciju leta sadrži naziv klase i jedan parametar:

`ServerSimulatoraLeta datoteka[.txt | .xml | .bin | .json]`

npr.

```
java -cp dist/dkermek_zadaca_1.jar org.foi.nwtis.dkermek.zadaca_1.ServerSimulatoraLeta
NWTiS_dkermek_zadaca_1.txt
```

Serverski program za simulaciju leta na početku provjerava postoji li datoteka konfiguracije te prekida rad ako ne postoji. Ako postoji, učitava postavke iz datoteke konfiguracije. Slijedi provjera da li su zauzeta zadana računalna vrata/port (postavka port.simulator). Ako su zauzeta, potrebno je ispisati poruku na konzolu i prekinuti rad programa. Ako nisu zauzeta, server kreira objekt za kolekciju letova aviona.

Sljedeći korak je priprema dretve za prijem zahtjeva korisnika. Kreira se objekt klase ServerSocket na zadanim računalnim vratima/port-u (postavka port.simulator) s maksimalnim brojem zahtjeva na čekanju (postavka maks.cekaca). Zatim se u petlji čeka da se spoji korisnik. Izlaz iz petlje dobije se prekidom rada od strane korisnika (crtl/c) ili sustava. Nakon spajanja korisnika, kreira se dretva koja obrađuje zahtjev.

Dretva iz dobivene veze na mrežnoj utičnici/socket-u 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 za server simulacije leta. Ako sintaksa primljenog zahtjeva nije ispravna tada se korisniku vraća odgovor ERROR 20; tekst; (tekst objašnjava razlog pogreške).

Server za simulaciju može primiti sljedeće komande od svog korisnika koji traži izvršavanja određene akcije:

- `LET icao24; POLIJETANJE yyyy.MM.dd HH:mm:ss; SLIJETANJE yyyy.MM.dd HH:mm:ss;`
  - server simulatora leta sprema informacije o avionu u svoju kolekciju letova. Ako ne postoji avion, on se samo dodaje u kolekciju letova. Ako postoji avion i on je sletio (trenutno vrijeme je veće od vremena sljetanja) tada se stari podaci brišu. Ako avion još leti (trenutno vrijeme je manje od vremena sljetanja) vraća se odgovor ERROR 21; tekst; (tekst objašnjava razlog pogreške). Ako je došlo do nekog drugog problema vraća se ERROR 22; tekst; (tekst objašnjava razlog pogreške). Ako je sve u redu, vraća odgovor OK;.
- `POZICIJA: icao24;`
  - server simulatora leta čita informacije o letu aviona. Ako postoje u kolekciji i avion još leti vraća OK; LETI; Ako je sletion vraća OK; SLETIO; Ako je došlo do problema vraća se ERROR 23; tekst; (tekst objašnjava razlog pogreške).

Pokretanje **serverskog programa** sadrži naziv klase i jedan parametar:

`ServerAviona datoteka[.txt | .xml | .bin | .json] --brojDretvi n`

npr.

```
java -cp dist/dkermek_zadaca_1.jar org.foi.nwtis.dkermek.zadaca_1.ServerAviona
NWTiS_dkermek_zadaca_1.txt --brojDretvi 7
```

Serverski program za rad s avionima na početku provjerava postoji li datoteka konfiguracije te prekida rad ako ne postoji. Ako postoji, učitava postavke iz datoteke konfiguracije. Slijedi provjera da li su zauzeta zadana računalna vrata/port (postavka port.avioni). Ako su zauzeta, potrebno je ispisati poruku na konzolu i prekinuti rad programa. Ako nisu zauzeta, provjerava se da li postoji datoteka sa podacima o korisnicima (postavka datoteka.korisnika) i datoteka sa podacima o aerodromima (postavka datoteka.aerodromi), ako ne postoji ili sadrži podatke u pogrešnom formatu potrebno je ispisati odgovarajuću poruku na konzolu i prekinuti rad programa.

Ako je sve u redu, server kreira objekt za kolekciju korisnika i objekt za kolekciju aerodroma i puni ih sa podacima iz odgovarajuće datoteke. Nakon toga server provjerava da li postoji datoteka sa serijaliziranim podacima aviona (postavka datoteka.avioni) te ju učitava ako postoji. Ako ne postoji, ispisuje informaciju na konzolu i nastavlja rad. Ako postoji ispisuje učitane avione (naziv aviona, aerodroma, vremena poljetanja i sljetanja) na konzolu i nastavlja rad.

Serverski program uspostavlja početno stanje na temelju postavki. Server kreira grupu dretvi pod nazivom {LDAP_korisničko_ime}_SD (SD = servisna dretva) koja služi za obavljanje serijalizacije i drugih servisnih poslova sustava. Za dretve iz te grupe mogu se koristiti klasa Timer, TimerTask i druge izvedene klase za upravljanje dretvama i sl. Nazivi dretvi dobiju se temeljem naziva grupe iza kojeg slijedi _{brojDretve}, gdje brojDretve predstavlja redni broj u grupi dretvi dobiven tijekom njenog kreiranja.

Potrebno je odabrati prikladnu klasu i kreirati objekt za kolekciju dretvi koje će biti u grupi dretvi. Prva SD zadužena je u svakom ciklusu za serijalizaciju podataka o avionima. Njen rad određen je postavkom interval.pohrane.aviona kojom se određuje pravilno vrijeme pokretanja ciklusa dretve. Postupak serijalizacije mora se provoditi međusobno isključivo u odnosu na druge dretve koje upisuju podatke o avionima.

Sljedeći korak je priprema za prijem zahtjeva korisnika. Prvo je potrebno odabrati prikladnu klasu i kreirati objekt za kolekciju dretvi koje će obrađivati zahtjeve korisnika. Slijedi kreiranje objekta klase ServerSocket na zadanim računalnim vratima/port-u (postavka port.avioni) s maksimalnim brojem zahtjeva na čekanju (postavka maks.cekaca). Zatim se u petlji čeka da se spoji korisnik. Izlaz iz petlje dobije se komandom za kraj rada. Nakon spajanja korisnika, ako ima raspoložive dretve (broj aktivnih dretvi manji je od vrijednosti parametra brojDretvi) kreira novu dretvu koju dodaje u kolekciju i povećava broj aktivnih dretvi. Ukoliko nema raspoložive dretve, korisniku se vraća odgovor ERROR 01; tekst; (tekst objašnjava razlog pogreške).

Ako ima raspoložive dretve, prosljeđuje joj se zahtjev korisnika. Dretva iz dobivene veze na mrežnoj utičnici/socket-u 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 primljenog zahtjeva nije ispravna tada se korisniku vraća odgovor ERROR 02; tekst; (tekst objašnjava razlog pogreške). Prilikom primitka svake komande korisnik se mora autenticirati. Dretva provjerava postoji li korisnik i njemu pridružena lozinka u kolekciji korisnika. Ako postoji, izvršava se dobivena komanda (opisana u posebnom dijelu). Kada nije u redu korisnik ili lozinka ne odgovara, vraća se odgovor ERROR 03; tekst;

(tekst objašnjava razlog pogreške). Server u slučaju primljene komande za završetak rada ili prekida rada od strane korisnika (Ctrl/C) ili sustava, treba obaviti serijalizaciju podataka o avionima prije stvarnog završetka rada. Dretva kada završi svoj rad smanjuje broj aktivnih dretvi te se briše iz kolekcije dretvi.

Server za rad s avionima može primiti sljedeće komande od svog korisnika koji traži izvršavanja određene akcije:

- `KORISNIK korisnik; LOZINKA lozinka; KRAJ;`
  - prekida prijem komandi i čeka da aktivne dretve završe s radom, serijalizira avione i završava rad. Korisniku se vraća odgovor OK;. Ako nešto nije u redu s prekidom rada ili serijalizacijom vraća se odgovor ERROR 11; tekst; (tekst objašnjava razlog pogreške).
- `KORISNIK korisnik; LOZINKA lozinka; DODAJ n;`
  - povećava maksimalni broj dretvi za n. Korisniku se vraća odgovor OK;. Ako nešto nije u redu s kreiranjem novih dretvi vraća se odgovor ERROR 12; tekst; (tekst objašnjava razlog pogreške).
- `KORISNIK korisnik; LOZINKA lozinka; UZLETIO icao24; POLAZIŠTE icaoP; ODREDIŠTE icaoO; TRAJANJE trajanjeLetaSekunde;`
  - čitaju se podaci o aerodromu polazišta i aerodromu odredišta. Ukoliko nije u redu s podacima aerodroma (npr. aerodrom ne postoji i sl.) vraća se odgovor ERROR 13; tekst; (tekst objašnjava razlog pogreške). Slijedi čitanje podataka aviona. Ako avion ne postoji dodaju se informacije u kolekciju aviona (vrijeme polijetanja jednako je trenutnom vremeno dok se vrijeme slijetanja izračunava na temelju dobivenog podatka trajanjeLetaSekunde i vremena polijetanja). Ako avion postoji i ako nisu isti podaci aoredroma odredišta i polazišta javlja se greška (ERROR 14; tekst; (tekst objašnjava razlog pogreške)). Ako avion postoji i podaci aerodroma su isti ili je dodan novi avion, šalje se komanda LET na server simulatora leta. Ako se od servera za simulator leta dobije odgovor OK; korisniku se vraća odgovor OK; UDALJENOST d; gdje je d udaljenost između dva aerodroma (cjelobrojna vrijednost). Ako je u odgovoru od servera simulatora leta dobiven ERROR, vraća se odgovor ERROR 15; tekst; (tekst objašnjava razlog pogreške).
- `KORISNIK korisnik; LOZINKA lozinka; ISPIS icao24;`
  - dohvaća informacije o avionu u kolekciji i zatim šalje komandu POZICIJA na server simulatora leta. Ako se od servera za simulator leta dobije odgovor koji započinje s OK; korisniku se prosljeđuje primljeni odgovor. Ako postoji problem (npr. nema aviona) vraća se odgovor ERROR 16; tekst; (tekst objašnjava razlog pogreške).

Pokretanje **korisničkog programa** sadrži naziv klase i parametre:

```
KorisnikAviona -k korisnik -l lozinka -s [ipadresa | adresa] -p port
        [--kraj | --dodajDretve n | --uzletio "AerodromPolazište:
    icaoP, AerodromOdredište: icaoO, Avion: icao24, trajanjeLeta: s"
    | --ispis icao24 ]
```

npr. pokretanje dodavanje aviona

```
java -cp dist/dkermek_zadaca_1.jar org.foi.nwtis.dkermek.zadaca_1.KorisnikAviona -k pkos -l
123456 -s localhost -p 9000 --uzletio "AerodromPolazište: LDZA, AerodromOdredište: EDDF,
Avion: avion864, trajanjeLeta: 7200"
```

npr. pokretanje ispis aviona

```
java -cp dist/dkermek_zadaca_1.jar org.foi.nwtis.dkermek.zadaca_1.KorisnikAviona -k pkos -l
123456 -s localhost -p 9000 --ispis avion864
```

Dozvoljene vrijednost za opcije:

- korisnik (min 3, maks 10 znakova) može sadržavati mala i velika slova, brojeve i znakove: _, -
- lozinka (min 3, maks 10 znakova) može sadržavati mala i velika slova, brojeve i znakove: -, #, !
- 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 9000 i 9999
- n je broj dretvi i može biti u intervalu između 1 i 20
- icao24 je ICAO24 kod aviona (npr. AVION-3c4b2e)
- icaoP i icaoO su ICAO kod aerodroma (npr. LDZA, JFK, EDDF)
- s je vrijeme trajanja leta u sekundama

Korisnički program može se izvršavati više puta kako bi se koristili različiti načini rada na bazi parametara. Izvršavanje korisničkog programa započinje utvrđivanjem vrste rada na bazi opcija u parametarima. Komunikacija korisničkog programa i servera temelji se na jednostavnom protokolu koji ima određenu sintaksu, a sadrži skup komandi. Komande i njihova sintaksa objašnjeni su ranije. Logično je da se prvo pokreće server, a nakon njega korisnici s različitim parametrima. Komunikaciju otvara korisnik tako da šalje zahtjev serveru u obliku određene komande. Ukoliko je server ugašen mora se ispisati odgovarajuća informacija na konzolu. Ako server radi on provodi analizu primljenog 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 klase koja je zadužena za program.

Korisnik servera za rad s avionima spaja se na server putem mrežne utičnice/socket-a i šalje komandu serveru na temelju upisanih parametara i traži izvršavanja određene akcije:

- upisan parametar *--kraj* – šalje se komanda:

```
KORISNIK korisnik; LOZINKA lozinka; KRAJ;
```

- upisan parametar *--dodajDretve* n – šalje se komanda:

```
KORISNIK korisnik; LOZINKA lozinka; DODAJ n;
```

- upisan parametar *--uzletio "AerodromPolazište: icaoP, AerodromOdredište: icaoO, Avion: icao24, trajanjeLeta: s"* - šalje se komanda

```
KORISNIK korisnik; LOZINKA lozinka;  UZLETIO icao24; POLAZIŠTE
icaoP; ODREDIŠTE icaoO;  TRAJANJE s;
```

- upisan parametar –-ispis icao24 – šalje se komanda:

```
KORISNIK korisnik; LOZINKA lozinka; ISPIS icao24;
```

Opisi komandi nalaze se u opisu servera za rad s avionima.

Zadaća_1: Višedretveni sustav "Informacije o aerodromima" putem mrežne utičnice/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 prethodne vježbe (vjezba_03_2.jar) u direktorij .\lib
5. Dodavanje Java biblioteke iz prethodne vježbe: desna tipka na mišu na Libraries / Add Jar/Forder / Odabrati .\lib\vjezba_03_2.jar
6. Dodati ostale potrebne biblioteke (gson)
7. Obrisati u Test Libraries biblioteke koje se odnose na JUnit verzije 5.*. Postaviti se mišem na Test Libraries i otvoriti sadržaj. Odabrati sve JUnit 5.* i desnom tipkom na mišu odabrati Remove.
8. Dodati u Test Libreries biblioteke za testiranje JUnit verzije 4.*. Postaviti se mišem na Test Libraries, aktivirati desnu tipku na mišu i odabratti Add Library, a zatim odabrati Harmcrest 1.3 i JUnit 4.1.2.
9. Preuzeti sa moodla datoteke postavki, datoteku sa podacima aerodroma i klasu **Koordinata**.
10. Kreiranje klase **Aerodrom**. Služi za evidenciju aerodroma koji se učitavaju iz datoteke kod pokretanja servera aviona.
11. Kreiranje klase **Avion**. Služi za evidenciju aviona 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 aviona i sl.
12. Kreiranje klase **Korisnik**. Služi za evidenciju podataka korisniku sustava. Treba odrediti varijable u koje će se pridružiti vrijednosti. Za sada samo se koriste korisničko ime i lozinka.
13. Kreiranje klase **ServisAviona** kao dretve ili druga prikladna klasa. Kreira se konstruktor klase u koji se prenose potrebni podaci za rad. Služi za serijalizaciju podataka aviona prema zadanom intervalu. Potrebno je voditi brigu o međusobnom isključivanju dretvi kod pristupa aerodromima i sl.
14. Kreiranje klase **ZahtjevAviona** kao dretve**.** Kreiranje konstruktora klase u koji se prenose potrebni podaci za rad. Dretva iz dobivene veze na mrežnoj utičnici/socket-u preuzima tokove za ulazne i izlazne podatke prema korisniku. Dretva preuzima podatke koje šalje korisnik putem ulaznog toka podataka, provjerava korektnost komande iz zahtjeva. Preporučuje se koristiti dopuštene izraze. Za prvo testiranje servera može se koristiti primjer s 4. predavanja Primjer33_3.java. Na kraju dretva vraća podatke korisniku putem izlaznog toka podataka. Za svaku vrstu komande kreira se posebna metoda koja odrađuje njenu funkcionalnost.
15. Kreiranje klase **KorisnikAviona**. To je izvršna klasa. Prvo se provjeravaju upisane opcije, preporučuje se koristiti dopuštene izraze. Objekt klase spaja se na server i šalje komandu u zahtjevu putem mrežne utičnice/socket-a. Primljeni odgovori se ispisuju na ekranu korisnika. Za svaku vrstu opcija kreira se posebna metoda koja odrađuje njenu funkcionalnost.
16. Kreiranje klase **ServerAviona i ServerSimulatoraLeta.** To su izvršne klase. Prvo se provjeravaju upisane opcije, preporučuje se koristiti dopuštene izraze. Za pomoć mogu poslužiti:
    - 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/
    - priloženi primjer uz zadaću (TestOpcija.java)
17. Učitavaju se postavke iz datoteke. Kreira se potrebna grupa dretvi. Kreiraju se i pokreću servisne dretve. Otvara se ServerSocket (slično primjerima ClientTester.java i TinyHttpd.java s 4. predavanja) koristeći konstruktor public ServerSocket(int port, int backlog) na izabranom portu uz maksimalni broj elemenata u redu čekanja na ostvarivanje veze. Zatim čeka zahtjev korisnika u beskonačnoj petlji. Kada se korisnik spoji na otvorenu vezu, provjerava raspoloživost dretvi za zahtjeve. Ako postoji slobodan broj kreira se dretva za obradu zahtjeva korisnika, kojoj se predaje veza i pokreće se izvršavanje dretve. Nakon toga server ponovno čeka na uspostavljanje veze i postupak se nastavlja. Dretva nakon obrade zahtjeva korisnika prestaje s radom.

18. Kreiranje JUnit klase **ServerAvionaTest ili ServerSimulatoraLetaTest** za testiranje klase ServerAviona ili ServerSimulatoraLeta. MIšem se postavlja na klasu ServerAviona ili ServerSimulatoraLeta desnom tipkom na mišu aktivira se izbornik u kojem se odabire Tools pa zatim Create/Update Tests. Zatim se odabire JUnit okvir (Framework) i postavljaju se svi elementi u dijelu za generiranje koda.
19. Ako se 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.