Diplomski studij

Informacijska i komunikacijska tehnologija
Telekomunikacije i informatika

Računarstvo
Računarska znanost
Programsko inženjerstvo i informacijski sustavi

Raspodijeljeni sustavi

Upute za izradu 1. domaće zadaće
Raspodijeljeno programiranje (TCP, UDP, RMI)

Ak. g. 2009./2010.

Sadržaj

Uvod

CILJ DOMAĆE ZADAĆE:

U praksi utvrditi i ponoviti gradivo s predavanja. Studenti će naučiti programirati 3 izvedbe raspodijeljenog sustava temeljenog na modelu klijent-poslužitelj. Kod prve izvedbe komunikacija će se odvijati protokolom UDP, kod druge protokolom TCP, a treća izvedba će biti poziv udaljene metode.

ZADATAK:

Ova domaća zadaća se sastoji od sljedeća 4 dijela:

- **proučavanje primjera s predavanja**,
- programiranje poslužitelja i klijenta koji komuniciraju protokolom UDP,
- programiranje poslužitelja i klijenta koji komuniciraju protokolom TCP i
- programiranje poslužitelja i klijenta koji komuniciraju pozivom udaljene metode.

Studenti trebaju programski izvesti 2 od 3 ponuđena zadatka, tj. jednu od sljedeće dvije kombinacije: a) programiranje UDP zadatka + programiranje TCP zadatka ili b) programiranje TCP zadatka + programiranje RMI zadatka.

PREDAJA:

Studenti su dužni u zadanom roku putem sustava *Moodle* predati arhivu koja se sastoji od sljedeća 4 dijela:

Kombinacija a)	Kombinacija b)
1. projekt UDP klijenta, 2. projekt UDP poslužitelja, 3. projekt TCP klijenta i 4. projekt TCP poslužitelja.	1. projekt TCP klijenta i 2. projekt TCP poslužitelja. 3. projekt RMI klijenta 4. projekt RMI poslužitelja

Navedene komponente trebaju biti realizirane u nekom od objektno-orijentiranih programskih jezika kao što su Java, C++, C# itd. Projekt je skup datoteka u odabranom razvojnom okružju (npr. Eclipse, Netbeans, Visual Studio, itd.).

Osim predaje samih datoteka u digitalnom obliku, bit će organizirana i usmena predaja na kojoj će ispitivati razumijevanje koncepata potrebnih za izradu domaće zadaće te poznavanje vlastitog programskog koda. Svi studenti trebaju proučiti primjere s predavanja, a moguće je da pri usmenoj predaji bude ispitivano i znanje studenta o primjeru s predavanja iz teme zadatka kojeg nisu odabrali.

Studenti koji budu kasnili s predajom će dobiti umanjeni broj bodova sukladno sljedećim kriterijima:

- 50% manje bodova za kašnjenje koje je kraće od tjedan dana i
- 100% manje bodova za kašnjenje koje je dulje od 1 tjedna.

1 Programiranje UDP klijenta i poslužitelja

U prvom dijelu domaće zadaće studenti trebaju maksimalno iskoristiti programski kod s predavanja te programski izvesti klijenta i poslužitelja koji pouzdano komuniciraju protokolom UDP. Protokol UDP ne osigurava pouzdanu komunikaciju od točke do točke, već je njena realizacija prepuštena višim slojevima (tj. aplikacijskom sloju). Kako će se i poslužitelj i klijent izvoditi na istom računalu te između njih neće biti stvarne mreže, potrebno je na neki način simulirati stvarnu mrežu u kojoj se paketi mogu izgubiti i stići različitim redoslijedom od redoslijeda slanja.

Za simuliranje komunikacije stvarnom mrežom potrebno je koristiti priloženu klasu `SimpleSimulatedDatagramSocket`. Ova klasa se koristi na način identičan onome klase `DatagramSocket`, a koji je bio objašnjen na predavanju. Razlika između ove dvije klase je u tome što klasa `SimpleSimulatedDatagramSocket` ima malo drugačiji konstruktor: `SimpleSimulatedDatagramSocket(double lossRate, int averageDelay)`. Ovaj konstruktor prima sljedeća 2 parametra: stopu gubitaka paketa u mreži [postotak] i prosječno kašnjenje paketa u jednom smjeru [milisekunde]. Ukoliko se student odluči za neki drugi programski jezik osim Jave, tada će morati simulirati stvarnu mrežu po uzoru na način ostvaren klasom `SimpleSimulatedDatagramSocket`.

Klijent treba zahtijevati dostavu odgovarajuće datoteke (npr. .wav, .mp3, .bin) od poslužitelja. Poslužitelj će datoteku podijeliti u nekoliko paketa, numerirati pakete i poslati ih klijentu. Klijent treba voditi brigu o redoslijedu primljenih paketa i sve dolazne pakete stavljati u spremnik. Kada klijent primi sve pakete, datoteku će pohraniti na tvrdi disk vašeg računala. Za svaki uspješno primljeni paket, klijent će poslužitelju poslati potvrdu (također u obliku paketa). Poslužitelj treba voditi brigu o izgubljenim paketima te ponoviti slanje svih izgubljenih paketa. Moguća je i situacija u kojoj će poneki paket biti poslan i po nekoliko puta.

Klasu `SimpleSimulatedDatagramSocket` treba koristiti prilikom slanja i s poslužiteljske i s klijentske strane te je stoga moguće i da neke potvrde budu izgubljene. Posebnu pozornost obratite na to da klijent i poslužitelj ne smiju biti realizirani u istom projektu odabrane razvojne okoline.

2 Programiranje TCP klijenta i poslužitelja

U drugom dijelu domaće zadaće potrebno je maksimalno iskoristiti programski kod s predavanja te programski izvesti klijenta i poslužitelja koji komuniciraju protokolom TCP. Podaci između klijenta i poslužitelja će se slati u obliku XML zapisa, a predstavljat će instance neke vaše klase.

Vaš zadatak je stvoriti klasu koja predstavlja neku grupu stvari, osoba ili predmeta iz stvarnog života, npr. a) student, b) automobil i c) nogometni klub. **Pri tome smislite neku svoju klasu i nemojte odabrati predložene klase koje su tu samo zbog ilustracije.** Ova klasa treba imati nekoliko atributa, od kojih poneki atributi trebaju također biti instance neke vaše druge klase, npr. za a) su to fakultet, prijatelj; za b) su to proizvođač, vlasnik; za c) su to menadžer, golman. Zbog toga je potrebno da

klijent i poslužitelj posjeduju logiku za čitanje i zapisivanje instanci vaše klase u obliku XML zapisa.
Klijent treba biti u stanju poslati i primiti (prikazati na zaslon) od poslužitelja instance vaše klase, dok poslužitelj treba pohranjivati primljene instance u obliku XML datoteke na tvrdi disk vašeg računala.
Posebnu pozornost obratite na to da klijent i poslužitelj ne smiju biti realizirani u istom projektu odabrane razvojne okoline. Pri tome projekti mogu imati neke zajedničke klase (npr. vaša klasa iz zadatka će biti dio i jednog i drugog projekta).

3 Programiranje RMI klijenta i poslužitelja

U zadnjem dijelu domaće zadaće maksimalno iskoristite programski kod s predavanja te programski izvedite klijenta i poslužitelja koji komuniciraju pozivom udaljene metode. Funkcionalnost klijenta i poslužitelja treba biti slična onoj u 2. dijelu domaće zadaće. Klijent treba pozivom udaljene metode na poslužitelju biti u stanju zatražiti i primiti (prikazati na zaslon) od njega instancu vaše klase. Za razliku od 2. dijela domaće zadaće, RMI klijent ne treba posjedovati funkcionalnost za slanje instanci vaše klase poslužitelju. Poslužitelj treba dohvaćati zapise (tj. instance) iz XML datoteke s tvrdog diska vašeg računala. Pri tome morate maksimalno iskoristiti vaše klase iz drugog dijela domaće zadaće. Dodatno, za svaku vašu klasu koja predstavlja podatak (vaša klasa i ostale klase čije instance predstavljaju atribute vaše klase) potrebno je napraviti (općenitije) sučelje.
Posebnu pozornost obratite na to da klijent i poslužitelj ne smiju biti realizirani u istom projektu odabrane razvojne okoline. Pri tome projekti mogu imati neka zajednička sučelja (npr. sučelja klasa koje predstavljaju podatke će biti dio i jednog i drugog projekta), ali niti jedna klasa koja predstavlja podatak ne smije biti dio klijentskog projekta. Klijent treba sve klase potrebne za komunikaciju s poslužiteljem dohvaćati putem Web poslužitelja Tomcat (ili nekog drugog, npr. Internet Information Services) kojeg trebate prethodno instalirati, podesiti i pokrenuti. Web poslužitelj će stoga obavljati ulogu poslužitelja klasa (koju je u primjeru s predavanja obavljala klasa `ClassServer`).
Redoslijed pokretanja RMI primjera je:

1. Pokretanje RMI registra:

   ```
   C:\Program Files\Java\jdk1.6.0_16\bin\rmiregistry.exe,
   ```

2. Pokretanje poslužitelja klasa (Tomcat, IIS),
3. Unošenje parametra Java virtualnih strojeva (u NetBeans – desni klik na projekt -> Properties -> Run -> VM Options):

   ```
   -Djava.rmi.server.codebase=http://localhost:4727/ -
   Djava.security.manager -Djava.security.policy=myclient.policy

   -Djava.rmi.server.codebase=http://localhost:4727/ -
   Djava.security.manager -Djava.security.policy=myserver.policy
   ```

4. Pokretanje RMI poslužitelja i
5. Pokretanje RMI klijenta.