Predavanje – Projektovanje softvera – Rad u mreži

2. Programiranje (rad) u mreži

- **Internet adresa**
- **147.91.128.47**
- **Povezivanje i prenos podataka između računara**
- **TCP/IP (Transimion Control Protocol/Internet Protocol)**
- **TCP – uspostavljanje i raskidanje veze, plus kontrolne funkcije**
- **IP – pronalaženje ciljnog računara i slanje podataka**
- **IP adresa**
- **Paket java.net**

```
// Primer  MR1: Prikazati Internet adresu tekuce masine.

import java.net.*;

class Mreza1
{
        public static void main(String args[]) throws UnknownHostException
          {
            InetAddress tekucaAdresa = InetAddress.getLocalHost();
            System.out.println(tekucaAdresa);
    }
}

// Rezultat:
// gsi/147.91.128.109
```

Simbolička adresa računara

- java.fon.bg.ac.yu
- DNS – povezivanje simboličkih i internet adresa

```
// Primer  MR2: Prikazati IP adresu masine koja ima simbolicku adresu "gsi.fon.bg.ac.yu"[1].

import java.net.*;

class Mreza2
{     public static void main(String args[]) throws UnknownHostException
            { InetAddress tekucaAdresa = InetAddress.getByName("java.fon.bg.ac.yu");
              System.out.println(tekucaAdresa); // Prikazuje simbolicku i IP adresu.
              System.out.println(tekucaAdresa.getHostAddress());// Prikazuje simbolicku adresu.
              System.out.println(tekucaAdresa.getHostName());// Prikazuje IP adresu.
     }}
// Rezultat:
// gsi.fon.bg.ac.yu/147.91.128.109
// 147.91.128.109
// gsi.fon.bg.ac.yu
```

Sledeći primer:

[1] Navedene internet i simboličke adrese su proizvoljne.

// **Primer MR3:** Prikazati IP adrese masina koje su vezane za simbolicku adresu: "www.nba.com"

```
import java.net.*;

class Mreza3
{
        public static void main(String args[]) throws UnknownHostException
          {
                InetAddress nizIPAdresa[] = InetAddress.getAllByName("www.nba.com");
                for(int i=0; i<nizIPAdresa.length;i++)
                  System.out.println(nizIPAdresa[i]);// Prikazuje IP adrese racunara.
    }
}
```

2.2 URL adresa

- **IP adrese i simboličke adrese – pristup do željenih računara u mreži**
- **URL (Uniform Resource Locator) – pristup do servisa i datoteka na računarima**
- **URL adresa sastoji se iz četiri dela:**
  a) **Protokol koji se koristi (http, ftp, gopher ili file).**
  b) **Adresa računara (internet ili simbolička adresa**
  c) **Broj porta.**
  d) **Putanja do datoteke, uključujući i ime datoteke.**

```
// Programski zahtev MR4: Prikazati svaki od delova URL adrese.
import java.net.*;
class Mreza4
{
        public static void main(String args[]) throws MalformedURLException
         {
                URL hp = new URL("http://java.fon.bg.ac.yu:80/index.htm");
                System.out.println("Protokol:" + hp.getProtocol());
                System.out.println("Port:" + hp.getPort());
                System.out.println("Racunar:" + hp.getHost());
                System.out.println("Datoteka:" + hp.getFile());
                System.out.println("Zajedno:" + hp.toExternalForm());
         }
}

// Rezultat:
// Protokol: http
// Port: 80
// Racunar: java.fon.bg.ac.yu
// Datoteka: /test.html
// Zajedno: http://java.fon.bg.ac.yu:80/index.htm
```

Sledeći primer:

```
// Primer MR5: Prikazati sadrzaj datoteke Pred.html kojoj se pristupa  pomocu URL adrese.


import java.net.*;
import java.io.*;
import java.util.Date;

class Mreza5
{       public static void main(String args[]) throws Exception
            {  int c;
               URL url = new URL("http://java.fon.bg.ac.yu/index.htm");
               URLConnection urlc = url.openConnection();

               System.out.println("Datum:" + new Date(urlc.getDate()));
              System.out.println("Vrsta sadrzaja:" + urlc.getContentType());
              System.out.println("Rok trajanja:" + urlc.getExpiration());
              System.out.println("Vreme zadnje izmene:" + new Date(urlc.getLastModified()));

              int duz = urlc.getContentLength();
              System.out.println("Duzina sadrzaja: " + duz);
              if (duz > 0)
               { InputStream is = urlc.getInputStream();
                  int i = duz;
                 while (((c = is.read()) != -1) && (--i > 0))   { System.out.println((char) c); }
                 is.close();
               }
             else
             { System.out.println("Nema podataka");}
             }
}
```

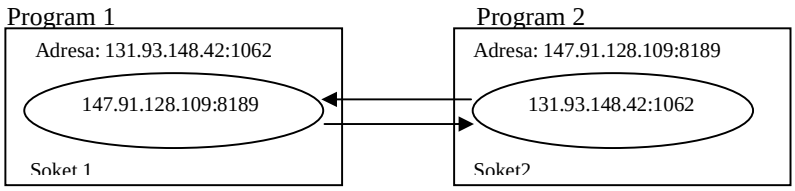
2.3 Soketi

- **Soket, u širem smislu, je mehanizam koji omogućava komunikaciju između programa koji se izvršavaju na različitim računarima u mreži**.
- **Pri povezivanju dva programa preko soketa, po jedan soket se generiše za svaki program. Svaki od soketa sadrži referencu na drugi soket. To praktično znači da prvi soket sadrži referencu na drugi soket, dok drugi soket sadrži referencu na prvi soket.**

2.3.1 Adresa soketa

Adresa soketa sastoji se iz dva dela:

a) adrese računara na kome se nalazi program koji je generisao soket
b) broja porta koji je generisan pomoću soketa



Sledeća strana:

Konekcija **između dva programa je ostvarena kada se uspostavi veza između njihovih soketa.**

- **Soketi, u užem smislu, predstavljaju objekte pomoću kojih se šalju/prihvataju podaci ka/od drugih soketa.**

- **Soket je po svojoj prirodi ulazno-izlazni tok i on se ponaša na sličan način kao:**
    1. **sistemski objekat System.in, pomoću koga se podaci prihvataju sa standardnog ulaza (tastature), dok se kod soketa podaci prihvataju sa spoljašnjeg ulaza (sa mreže) od drugog soketa.**
    2. **sistemski objekat System.out, pomoću koga se podaci šalju ka standardnom izlazu (ekranu), dok se kod soketa podaci šalju ka spoljašnjem izlazu (ka mreži) do drugog soketa.**

- **Soketi se povezuju sa ulazno-izlaznim tokovima na sličan način kao što je to slučaj sa System.in i System.out objektima, kada se želi izvršiti obrada podataka koju soketi razmenjuju.**

Sledeća strana:

Scenario K/S aplikacija

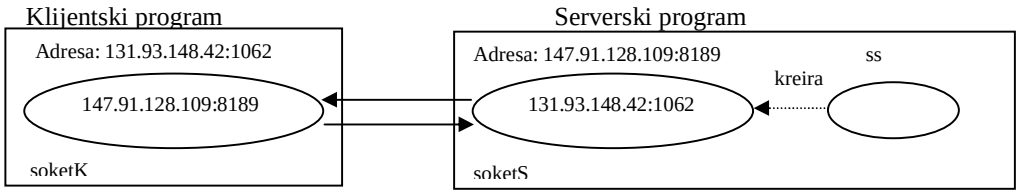
Server

- **Pokreće se program na serverskom računaru. Internet adresa server računara je 147.91.128.109.**
- ***ServerSocket ss = new ServerSocket(8189);* pravi se server soket koji se povezuje npr. sa portom 8189.**
- **Adresa server soketa *ss* je 147.91.128.109:8189.**
- ***Socket soketS = ss.accept();***

Klijent

- ***Socket soketK = new Socket("147.91.128.109",8189);***
- **Soket koji je napravljen na klijentskoj strani prosleđuje do serverskog soketa svoju adresu: *131.93.148.42:1062*.**

.



Sledeća strana:

Povezivanje soketa sa UI tokovima

- ***BufferedReader in = new BufferedReader(new InputStreamReader(X.getInputStream()));***

  PrintWriter out = new PrintWriter(X.getOutputStream(),true);

  X ∈ (soketS, soketK)

- ***out.println(" Y je spreman za rad\n");***

  Y ∈ (KLIJENT, SERVER)

- **Klijent, odnosno server prihvata podatke na sledeći način:**

  String line =in.readLine();

- **Na kraju klijent, odnosno server na standardnom izlazu prikazuju poruku koju je primio:**
  System.out.println(" Z je primio poruku od Z1:" + line);
  Z ≠ Z1
  Z , Z1∈ *(KLIJENT, SERVER)*

- **Soketi su ravnopravni u komunikaciji.**

Primer:

MR6S

Serverski program

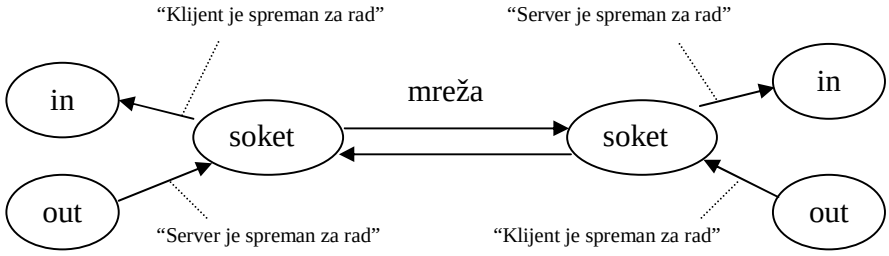
```
// Primer MR6S:: Napisati program koji ce kreirati serverski soket na portu 8189.
// Nakon toga se povezati sa klijentskim soketom. Na kraju poslati poruku klijentskom
// soketu.

import java.io.*;
import java.net.*;

public class ServerSoket
{  public static void main(String[] args)
    {  try
      {  ServerSocket ss = new ServerSocket(8189);
         Socket soketS = ss.accept();
        BufferedReader in  = new BufferedReader(new InputStreamReader(soketS.getInputStream()));
        PrintWriter out = new PrintWriter(soketS.getOutputStream(),true);
        out.println(" SERVER je spreman za rad\n");
        String line = in.readLine();
        System.out.println(" SERVER je primio poruku od klijenta:" + line);
      }  catch (Exception e)     { System.out.println(e);}
 }}
```

Klijentski program:

MR6K

Klijentski program

```
// Primer MR6K:: Napisati program koji ce kreirati klijentski soket, koji ce se
// povezati sa serverskim soketom koji je podignut na racunaru cija je adresa
// 147.91.128.109 na portu 8189.Poslati poruku serverskom racunaru.
import java.io.*;
import java.net.*;
public class SoketKlijent
{  public static void main(String[] args)
  {  try
      { String s;
        Socket soketK = new Socket("147.91.128.109",8189);
       BufferedReader in = new BufferedReader(new InputStreamReader(soketK.getInputStream()));
       PrintWriter out = new PrintWriter(soketK.getOutputStream(),true);
       out.println(" KLIJENT je spreman za rad\n");
       String line =in.readLine();
       System.out.println(" KLIJENT je primio poruku od servera:" + line);
     }  catch (Exception e)
            { System.out.println(e);}
 }}
```



Sledeća strana:

Adrese klijentskog i serverskog programa

- **Ukoliko se želi videti port soketa na koji pokazuje drugi soket, koristi se metoda *getPort()*.**
- **IP adresa soketa na koji pokazuje drugi soket, koristi se metoda *getInetAddress()*.**
- **lokalni port na kome je podignut soket koristi se metoda *getLocalPort()*. To znači da se puna adresa soketa na koji pokazuje drugi soket dobija kao: *getInetAddress()* + *getPort()* što se može videti u sledećem primeru:**

Serverski program

```
// Primer MR7S: Prikazati IP adresu racunara i broj porta na kome se nalazi
// klijentski soket. Na kraju prikazati broj porta na kome se nalazi serverski soket.

import java.io.*;
import java.net.*;

public class ServerSoket
{  public static void main(String[] args)
   {   try
         {  ServerSocket ss = new ServerSocket(8189);
            System.out.println("SERVER");
            Socket soketS = ss.accept();
           // ia dobija IP adresu racunara na kome se nalazi klijentski soket
           InetAddress ia = soketS.getInetAddress();
           // getPort() metoda prikazuje port na kome se nalazi klijentski soket
           System.out.println(ia + " " + soketS.getPort());
          // getLocalPort() metoda prikazuje port na kome se nalazi serverski soket (8189)
          System.out.println(soketS.getLocalPort());
       }  catch (Exception e)   { System.out.println(e);}
  }
}
```

Klijentski program:

Klijentski program – PRMR7K

```
// PRMR7K: Napisati program koji ce kreirati klijentski soket koji ce se
// povezati sa serverskim soketom koji je podignut na lokalnom racunatu na portu 8189.
// Prikazati IP adresu racunara i broj porta na kome se nalazi serverski soket.
// Na kraju prikazati broj porta na kome se nalazi klijentski soket.


import java.io.*;
import java.net.*;

public class SoketKlijent
{  public static void main(String[] args)
  {    try   { String s;
               Socket soketK = new Socket("147.91.128.109",8189);
              InetAddress ia = soketK.getInetAddress();
             System.out.println(ia + " " + soketK.getPort() + " " + soketK.getLocalPort());
              } catch (Exception e)    { System.out.println(e);}
  }}
```

2.3.3 Povezivanje servera sa više klijenata

- **Soketi omogućavaju da se više klijentskih programa (klijent soketa) poveže na jedan serverski program (serverski soket). Za svaki od klijentskih soketa pravi se po jedna nit, tako da se u okviru serverskog programa konkurentno izvršava više niti. Navedene niti mogu da pristupe zajedničkim resursima servera.**

Server i TelNet programi (klijenti)

```
// Primer MR8S: Napisati program koji ce kreirati serverski soket na portu 8189. Serverski
// soket moze da se  poveze sa najvise 10 klijenata (klijentskih soketa). Za svakog
// klijenta napraviti posebnu nit koja ce se nezavisno izvrsavati u odnosu na druge niti.
// U okviru svake niti ce se vrsiti obrada kolicine robe (prodaja i nabavka).
// Kolicina robe ce biti zajednicki atribut svih klijenata.

import java.io.*;
import java.net.*;

public class ObradaRobe
{  public static void main(String[] args)
    {    try  {  KreiranjeNiti kn = new KreiranjeNiti();
                 kn.Kreiranje();
              }  catch (Exception e)   { System.out.println(e);}
   }
}

class KreiranjeNiti
{   int kolicina;
    ObradaNiti on[];
    ServerSocket ss;
    KreiranjeNiti() { on = new ObradaNiti[10];}
    public void Kreiranje()
    {         try {  ss = new ServerSocket(8189);
                      kolicina=10;
                     for (int brojKlijenta = 0;brojKlijenta < 10;brojKlijenta ++)
                   { Socket soketS = ss.accept();
                     System.out.println("Klijent " + brojKlijenta);
                     on[brojKlijenta] = new ObradaNiti(soketS,brojKlijenta,this);
                     on[brojKlijenta].start();
                    }
                  } catch (Exception e)  { System.out.println(e + " greska!");}
    }
}

class ObradaNiti extends Thread
{ public ObradaNiti(Socket soketS1,int c,KreiranjeNiti kn1)
   { soketS = soketS1; brojKlijenta=c+1; kn = kn1;}

  public void run()
   { try {  in = new BufferedReader(new InputStreamReader(soketS.getInputStream()));
            out = new PrintWriter(soketS.getOutputStream(),true);
            boolean done = false;
            while (!done)
            { out.println("Izaberite jednu od sledecih opcija:\n");
```

```
            out.println("1.PRODAJA. 2.NABAVKA 3. IZLAZ");
            out.println("  ");
           String line = in.readLine();
           if (line == null) done = true;
          else
            { switch (line.charAt(0))
               {  case '1': out.println("Klijent: ("+ brojKlijenta + "): IZABRANA PRODAJA" );
                          if (kn.kolicina == 0)  { out.println("Nema robe na zalihama!");}
                         else  { kn.kolicina = kn.kolicina -1;} break;
                  case '2': out.println("Klijent: ("+ brojKlijenta + "): IZABRANA NABAVKA" );
                            kn.kolicina = kn.kolicina+1 ;break;
                  default : done = true;
                }
             }
         out.println("Ukupno je ostalo komada:" + kn.kolicina);
       }
       soketS.close();
    }   catch (Exception e)  { System.out.println(e);  }
  }
  private Socket soketS;
  private int brojKlijenta;
  private KreiranjeNiti kn;
  BufferedReader in;  PrintWriter out;
 }
```

Detaljan opis programa:

Detaljni opis programa MR8S:

Server

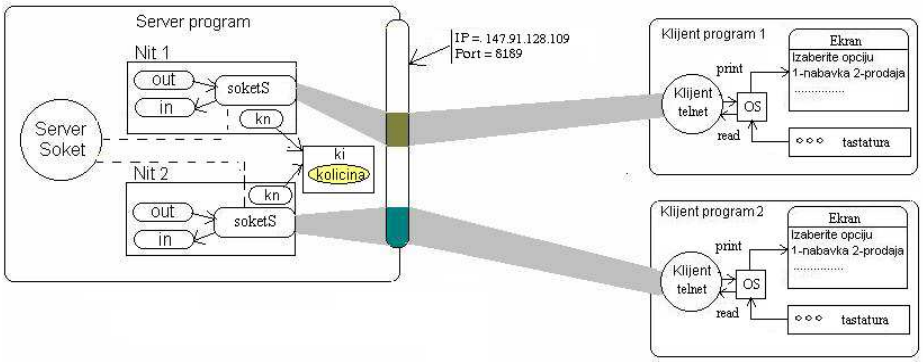
- ***ServerSocket ss = new ServerSocket(8189);***
- ***Socket soketS = ss.accept();***

Klijent

- ***Socket socket = new Socket(147.91.128.109,8189) ;***

Server

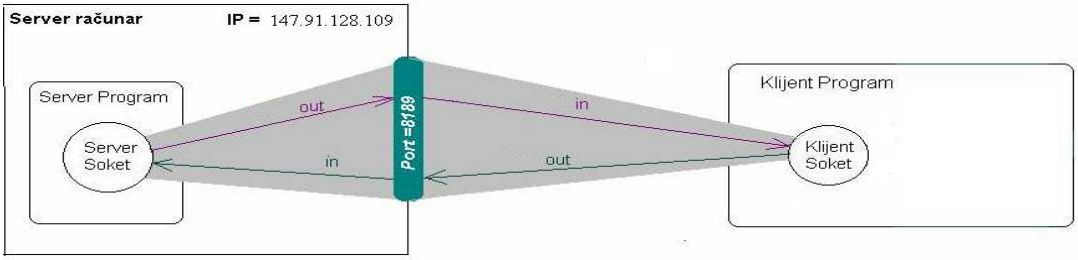
- ***on[brojKlijenta] = new ObradaNiti(soketS,brojKlijenta,this);***



Sledeća strana:

- ***Pokreće se run() metoda novokreirane niti.***

- ***BufferedReader in*** *= nesw BufferedReader(new InputStreamReader (soketS.getInputStream ()));*
  PrintWriter out *= new PrintWriter(soketS.getOutputStream(),true);*



- **Naredbe kao što su:**
  out.println("Izaberite jednu od sledecih opcija:\n");
  out.println("1.PRODAJA. 2.NABAVKA 3. IZLAZ");
  …
  out.printl(…)
  prikazuju navedene sadržaje na ekranu preko telnet programa.

- **Naredba *out.println()* server šalje podatke do klijenta (telnet) .**
- **Naredba *in.readLine()* - server prihvata podatke od klijenta.**

Server i Java programi (klijenti)

```
// Primer MR9K: Napisati program koji ce kreirati klijentski soket koji ce se
// povezati sa serverskim soketom koji je podignut na racunaru cija je IP adresa
// 147.91.128.109 na portu 8189.

import java.io.*;
import java.net.*;

public class SoketKlijent1
{
 public static void main(String[] args)
  {    try   { String s,line;
                Socket soketK = new Socket("147.91.128.109",8189);
                BufferedReader in = new BufferedReader(new InputStreamReader(soketK.getInputStream()));
                PrintWriter out = new PrintWriter(soketK.getOutputStream(),true);
                BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
                while(true)
                 { line = in.readLine();
                   // Kada serverski program posalje "999" prekinuce se izvrsenje klijenta.
                   if (line.equals("999"))
                     { soketK.close();   break;
                     }
                   System.out.println(line);
                  // Kada serverski program posalje "  " , klijent dobija mogucnost da izabere opciju.
                   if (line.equals("  "))
                     { s=br.readLine();
                       out.println(s);
                     }
                 }
          }  catch (Exception e)   { System.out.println(e);}
  }
}
```

Ukoliko se javi potreba da serverski program posalje klijentima obavestenje, ukoliko vrednost kolicine padne na 0, tada serverski program ima sledeći izgled:

```
// Primer MR10S: Napisati program koji ce kreirati serverski soket na portu 8189. Serverski
// soket moze da se  poveze sa najvise 10 klijenata (klijentskih soketa). Za svakog
// klijenta napraviti posebnu nit koja ce se nezavisno izvrsavati u odnosu na druge niti.
// U okviru svake niti ce se vrsiti obrada kolicine robe (prodaja i nabavka).
// Kolicina robe ce biti zajednicki atribut svih klijenata.
// Kada roba padne na koliciju jednaku 0 server treba da o tome obavesti sve klijente.



import java.io.*;
import java.net.*;

public class ObradaRobe1
{  public static void main(String[] args)
   {    try   { KreiranjeNiti kn = new KreiranjeNiti();
                kn.Kreiranje();
              }  catch (Exception e)  { System.out.println(e);}
  }
```

```
}

class KreiranjeNiti
{   int kolicina;
    ObradaNiti on[];
    ServerSocket ss;
    int brojKlijenta;

    KreiranjeNiti() { on = new ObradaNiti[10];}
    public void Kreiranje()
     {  try  {  ss = new ServerSocket(8189);
                kolicina=10;
                for (brojKlijenta = 0;brojKlijenta < 10;brojKlijenta++)
                   {  Socket soketS = ss.accept();
                     System.out.println("Klijent " + brojKlijenta);
                     on[brojKlijenta] = new ObradaNiti(soketS,brojKlijenta,this);
                     on[brojKlijenta].start();
                   }
             }  catch (Exception e){ System.out.println(e + " greska!");}
     }

    public void Prodaja(float p)
    {  kolicina-=p;
       if (kolicina<=0) Azuriranje();
    }

   public void Nabavka(float n)
   { kolicina+=n;    }

    public void Azuriranje()
    {  NitObavestenje obavestenje=new NitObavestenje(this,"Magacin se upravo izpraznio !");
       obavestenje.start();
      System.out.println("kreirao nit obavestenje");
   }
}

class ObradaNiti extends Thread
{ public ObradaNiti(Socket soketS1,int c,KreiranjeNiti kn1)
   { soketS = soketS1; brojKlijenta=c+1; kn = kn1;}

   public void run()
   {   try   {     in = new BufferedReader(new InputStreamReader(soketS.getInputStream()));
                   out = new PrintWriter(soketS.getOutputStream(),true);
                   boolean done = false;
                   while (!done)
                    { out.println("Izaberite jednu od sledecih opcija:\n");
                      out.println("1.PRODAJA. 2.NABAVKA 3. IZLAZ");
                      out.println("  ");
                      String line = in.readLine();
                      if (line == null  || line.equals(""))
                          { out.println("Zavrsetak rada klijenta");
                            out.println("999");
                            done = true;
                          }
                      else
                       { switch (line.charAt(0))
```

```
                    {   case '1': out.println("Echo: ("+ brojKlijenta + "): IZABRANA PRODAJA" );
                                        if (kn.kolicina - 4 < 0)
                                          { out.println("Nema robe na zalihama!");}
                                        else
                                          {kn.Prodaja(4);} break;
                        case '2': out.println("Echo: ("+ brojKlijenta + "): IZABRANA NABAVKA" );
                                  kn.Nabavka(2); break;
                         case '3': out.println("Zavrsetak rada klijenta");
                                  out.println("999");
                                  done = true;
                    }
                  }
        out.println("Ukupno je ostalo komada:" + kn.kolicina);
       }
       soketS.close();
       System.out.println("Zatvorio klijenta "+ brojKlijenta);
    }

   catch (Exception e)
    { System.out.println(e);
    }
 }
 private Socket soketS;
 public int brojKlijenta;
 private KreiranjeNiti kn;
 BufferedReader in;
 PrintWriter out;
}

class NitObavestenje extends Thread
{  public NitObavestenje(KreiranjeNiti k,String o)
   { kn=k;
     obavestenje=o;
   }

 private KreiranjeNiti kn;

 private String obavestenje;
 public void run()
 {  for(int i=0;i<kn.brojKlijenta;i++)
     {  try { kn.on[i].out.println("Echo("+kn.on[i].brojKlijenta+") "+obavestenje);
              System.out.println("Obavestio klijenta "+(i+1));
            }
        catch (Exception e){  System.out.println("Nema klijenta "+(i+1));   }
   }
  }
}
```

```
// Primer MR11K: Napisati program koji ce kreirati klijentski soket koji ce se
// povezati sa serverskim soketom koji je podignut na racunatu cija je IP adresa
// 147.91.128.109 na portu 8189.

import java.io.*;
import java.net.*;

public class SoketKlijent3
{  public static void main(String[] args)
   {    try { String s;
               Socket soketK = new Socket("127.0.0.1",8189);
               BufferedReader in = new BufferedReader(new InputStreamReader(soketK.getInputStream()));
               PrintWriter out = new PrintWriter(soketK.getOutputStream(),true);
               BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
                boolean signal = true;
             NitKlijent nk = new NitKlijent(in);
             nk.start();
             // Prihvata preko tastature podatke
             //s=br.readLine();
            //out.println(s);
            while(true)
            { //Prihvata preko tastature podatke
              s=br.readLine();
              // Salje podatke do servera
              out.println(s);
             }
       }  catch (Exception e)  { System.out.println(e);}
  }
}


class NitKlijent extends Thread
{ NitKlijent(BufferedReader in1) { in = in1;signal = true;}
  public void run()
    { try {  while(signal)
               {  // Prihvata podatke od servera
                  String line = in.readLine();
                 if (line.equals("999"))
                    break;
                // Prikazuje podatke na monitoru
                 System.out.println(line);
               }
         } catch(Exception e) {System.out.println("Lose primeljena poruka od servera!");}
  }

 void Prekini() { signal = false;}
 boolean signal = true;
 BufferedReader in;
}
```

```
// Primer MR11S: Napraviti serverski soket koji ce da posreduju u komunikaciji
// izmedju najvise 10 klijenata.


import java.io.*;
import java.net.*;

public class Chat
{  public static void main(String[] args)
  {    try  {  KreiranjeNiti ki = new KreiranjeNiti();
                ki.Kreiranje();
            }  catch (Exception e)   { System.out.println(e);}
  }
}

class KreiranjeNiti
{   int kolicina;
    int BrojKlijenata;
    NitProdaje[] np = new NitProdaje[10]; //Moze najvise 10 klijenata da se poveze sa serverom.
    public void Kreiranje()
     {  System.out.println("SERVER JE SPREMAN ZA RAD!!!\n");
        try  {  ServerSocket ss = new ServerSocket(8189);
                kolicina=10;
               for (BrojKlijenata = 0;BrojKlijenata<10;BrojKlijenata++)
                   {  Socket soketS = ss.accept();
                      System.out.println("Klijent " + (BrojKlijenata+1));
                      np[BrojKlijenata]=new NitProdaje(soketS,BrojKlijenata,this);
                      np[BrojKlijenata].start();
                   }
            }   catch (Exception e) { System.out.println(e);}
     }

    public void PosaljiSvimaPoruku(String poruka,int bklijenta)
    { NitObavestenje obavestenje=new NitObavestenje(this,poruka,bklijenta);
      obavestenje.start();
    }
}

class NitProdaje extends Thread
{ public NitProdaje(Socket soketS1,int brojKlijenta1,KreiranjeNiti kn1)
   { soketS = soketS1; brojKlijenta=brojKlijenta1+1; kn = kn1;
     try{ in = new BufferedReader(new InputStreamReader(soketS.getInputStream()));
          out = new PrintWriter(soketS.getOutputStream(),true);
        } catch(Exception e) { System.out.println(e);}
   }

  public void run()
   {  try {  out.println("SERVER U PRIPREMI POSREDOVANJA U RAZGOVORU:\n");
             out.println("Unesi svoje ime:\n");
             ime = in.readLine();
             out.println("Potvrdjujemo unos imena:\n\n" + ime);
             out.println("SERVER JE SPREMAN ZA POSREDOVANJA U RAZGOVORU:\n");
             while (true)
              { String line = in.readLine();
                kn.PosaljiSvimaPoruku(line,brojKlijenta);
             }
```

```
      } catch (Exception e){ System.out.println(e);}
  }

  public Socket soketS;
  public int brojKlijenta;
  private KreiranjeNiti kn;
  BufferedReader in;
  PrintWriter out;
  String ime;
 }

class NitObavestenje extends Thread
{  public NitObavestenje(KreiranjeNiti k,String o, int bklijenta1)
   { kn=k;
     obavestenje = o;
    bklijenta = bklijenta1-1;
   }

 private String obavestenje;
 public void run()
 { for(int i=0;i<kn.BrojKlijenata;i++)
   { try { kn.np[i].out.println(kn.np[bklijenta].ime + " :" + obavestenje);
         }  catch (Exception e){  System.out.println("Nema klijenta "+(i+1));}
   }
  }

  private KreiranjeNiti kn;
  int bklijenta;
}

// Primer MR11K: Napisati program koji ce kreirati klijentski soket koji ce se
// povezati sa serverskim soketom koji je podignut na racunatu cija je IP adresa
// 147.91.128.109 na portu 8189. Serverski soket treba da omoguci menjusobnu razmenu
// poruka vise klijenata.

import java.io.*;
import java.net.*;

public class SoketKlijent1
{  public static void main(String[] args)
   {    try { String s;
               Socket soketK = new Socket("147.91.128.109",8189);
              BufferedReader in = new BufferedReader(new InputStreamReader(soketK.getInputStream()));
              PrintWriter out = new PrintWriter(soketK.getOutputStream(),true);
             BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
             boolean signal = true;
             NitKlijent nk = new NitKlijent(in);
             nk.start();

             s=br.readLine();
            out.println(s);
            while(true)
             { s=br.readLine();
               out.println(s);
              }
        }  catch (Exception e)  { System.out.println(e);}
   }
}
```

```
class NitKlijent extends Thread
{ NitKlijent(BufferedReader in1) { in = in1;signal = true;}
   public void run()
     { try {  while(signal)
                 {  String line = in.readLine();
                    System.out.println(line);
                 }
          } catch(Exception e) {System.out.println("Lose primeljena poruka od servera!");}

   }

  void Prekini() { signal = false;}
  boolean signal = true;
  BufferedReader in;
}
```