SwRTc – Proiect

Procesul dezvoltarii unui sistem software orientat spre obiecte (II)

P.2. Utilizarea diagramelor UML ca suport pentru procesul de dezvoltare a unui sistem software (II)

P.2.1. Evaluarea versiunii anterioare a sistemului software OO

Dezvoltarea sistemelor software complexe este de obicei realizata in mai multe iteratii in care:

- **sunt reluate activitatile de analiza, proiectare, implementare, testare,**
- **sunt adaugate activitatile** de
  - **evaluare a iteratiei anterioare** si de
  - **planificare a iteratiilor ulterioare.**

Rezultatele iteratiei anterioare sunt fie **o versiune a sistemului software**, fie doar **un prototip al acestuia.**

Evaluarea rezultatelor iteratiei anterioare consta, printre altele, din:

- **verificarea functionarii corecte a sistemului** (conformitatii cu specificatiile),
- evaluarea documentatiei si a codului (analiza codului),
- **evaluarea gradului in care versiunea obtinuta corespunde asteptarilor utilizatorilor**.

In ceea ce priveste <u>evaluarea sistemului de comunicatie pentru conversatie textuala</u>, <u>dezvoltat in iteratia anterioara</u>, pot fi facute, printre altele, urmatoarele observatii:

- interfata grafica este rudimentara, modul de conectare la server putand fi imbunatatit,

- utilizatorii sunt identificati pe baza numelui de utilizator de pe masina pe care ei lanseaza componentele client, in loc sa isi poata alege un nume de utilizator al sistemului de conversatie (*nickname*)

- bara de defilare a zonei de text ar fi preferabil sa fie in mod automat actualizata astfel incat sa fie vizibile intotdeauna ultimele mesaje primite,

- utilizatorii nu au nici o informatie despre ceilalti utilizatori prezenti in conversatie

- pe un server poate exista doar un singur grup de utilizatori la un moment dat (alternativa ar fi posibilitatea crearii unor "camere", grupuri, etc.)

- nu exista posibilitatea crearii unor canale de comunicatie private, pentru conversatia intre 2 utilizatori

- etc. (<u>lista va fi largita prin discutii la laborator/proiect</u>).

P.2.2. Actualizarea cerintelor sistemului

In urma acestei evaluari, **presupunem ca s-a decis ca in iteratia urmatoare sa fie realizate urmatoarele imbunatatiri**:

- posibilitatea oferita utilizatorilor de a-si alege un nume de utilizator (*nickname*)

- actualizarea automata a barei de defilare a zonei de text pentru ca ultimele mesaje primite sa fie vizibile

- difuzarea de catre server a numelor de utilizator catre toti utilizatorii si informarea de catre client a utilizatorilor despre modificarile in componenta grupului de utilizatori

Desigur, raman nerezolvate anumite posibile cerinte ale utilizatorilor, cum ar fi

- posibilitatea crearii mai multor "camere" sau grupuri pe un server

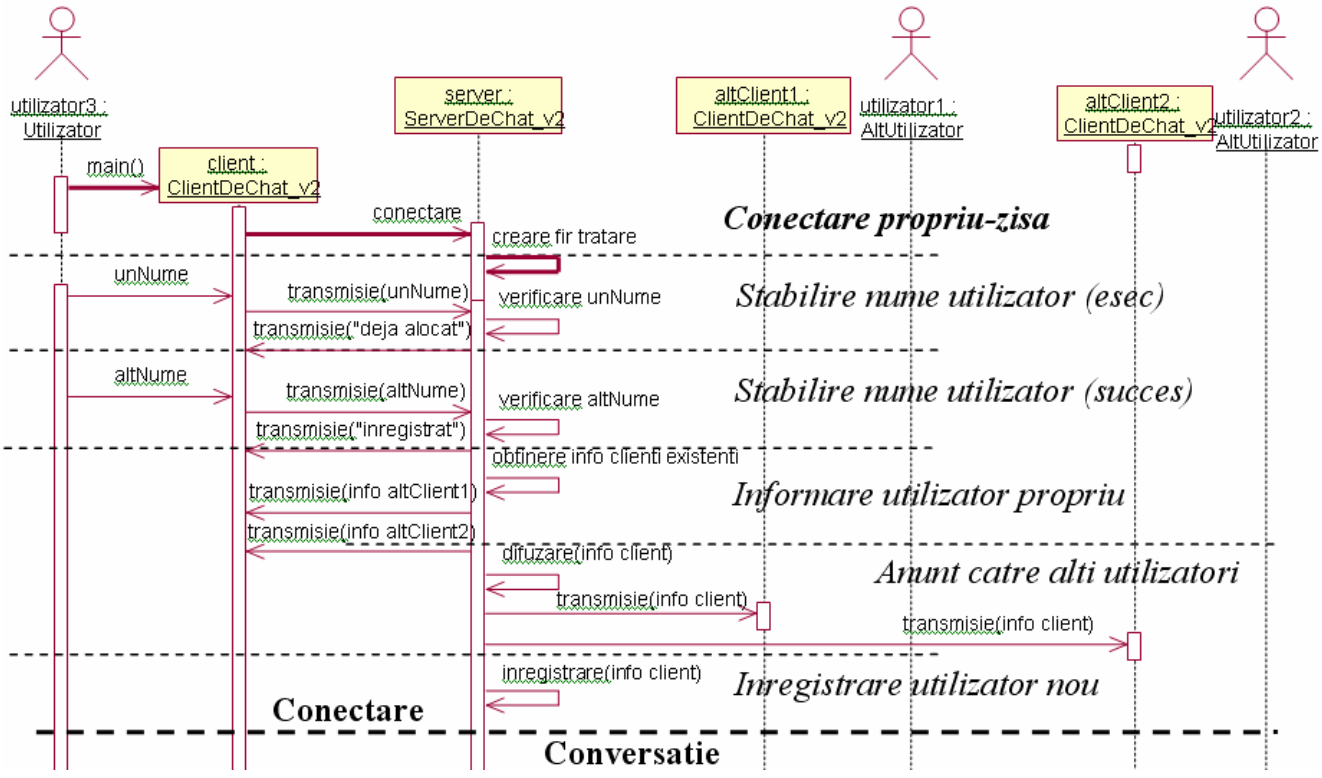
- posibilitatea crearii unor canale de comunicatie private

- etc **(lista va fi largita prin discutii la laborator/proiect)**.

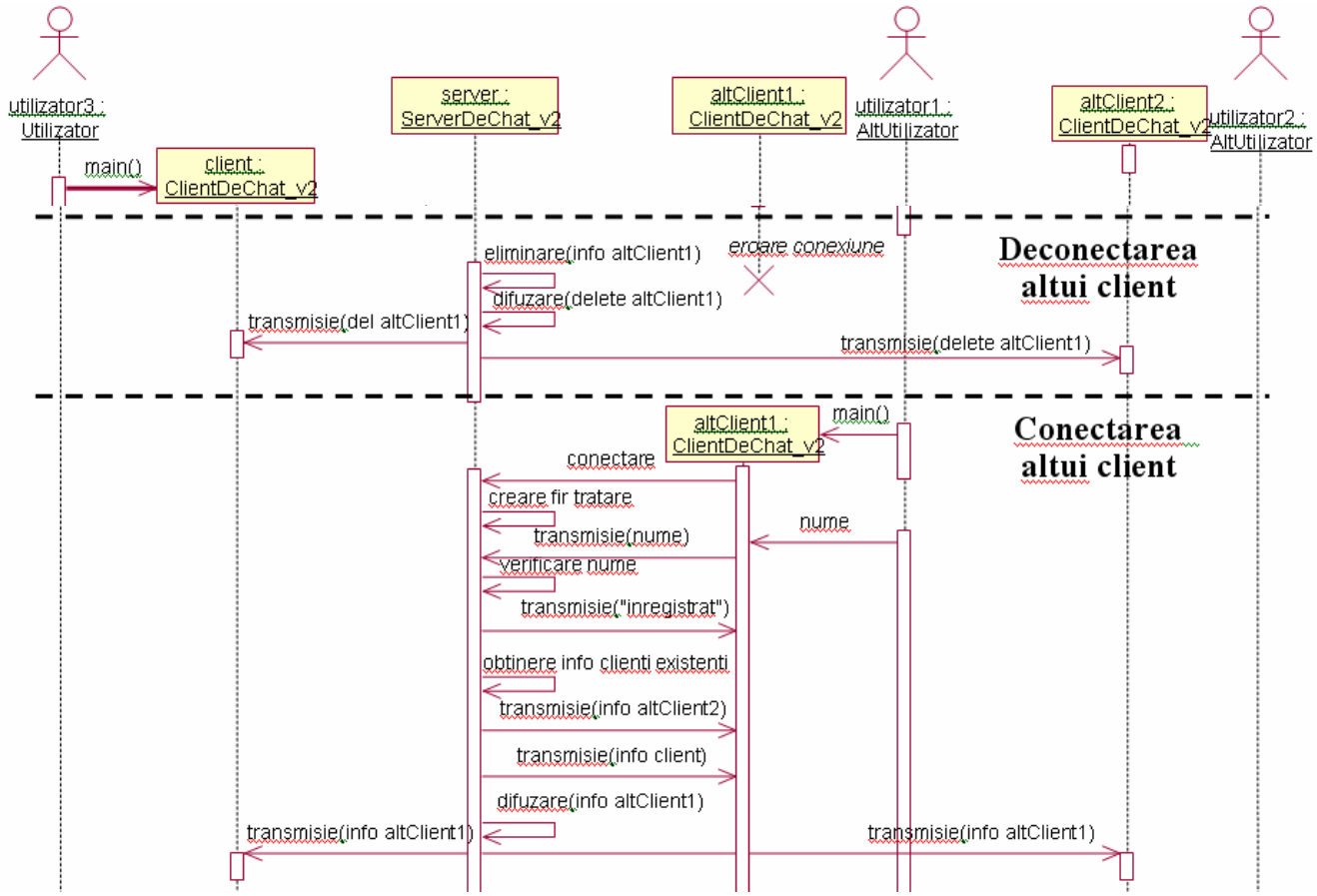
P.2.3. Actualizarea analizei OO a sistemului

Actualizarea analizei OO presupune:
- actualizarea diagramei UML a cazurilor de utilizare,
- actualizarea descrierii cazurilor de utilizare
- actualizarea listei claselor principale ale sistemului
- actualizarea diagramelor UML de secventa care descriu scenariile componente ale cazurilor de utilizare
- actualizarea diagramelor UML de colaborare corespunzatoare

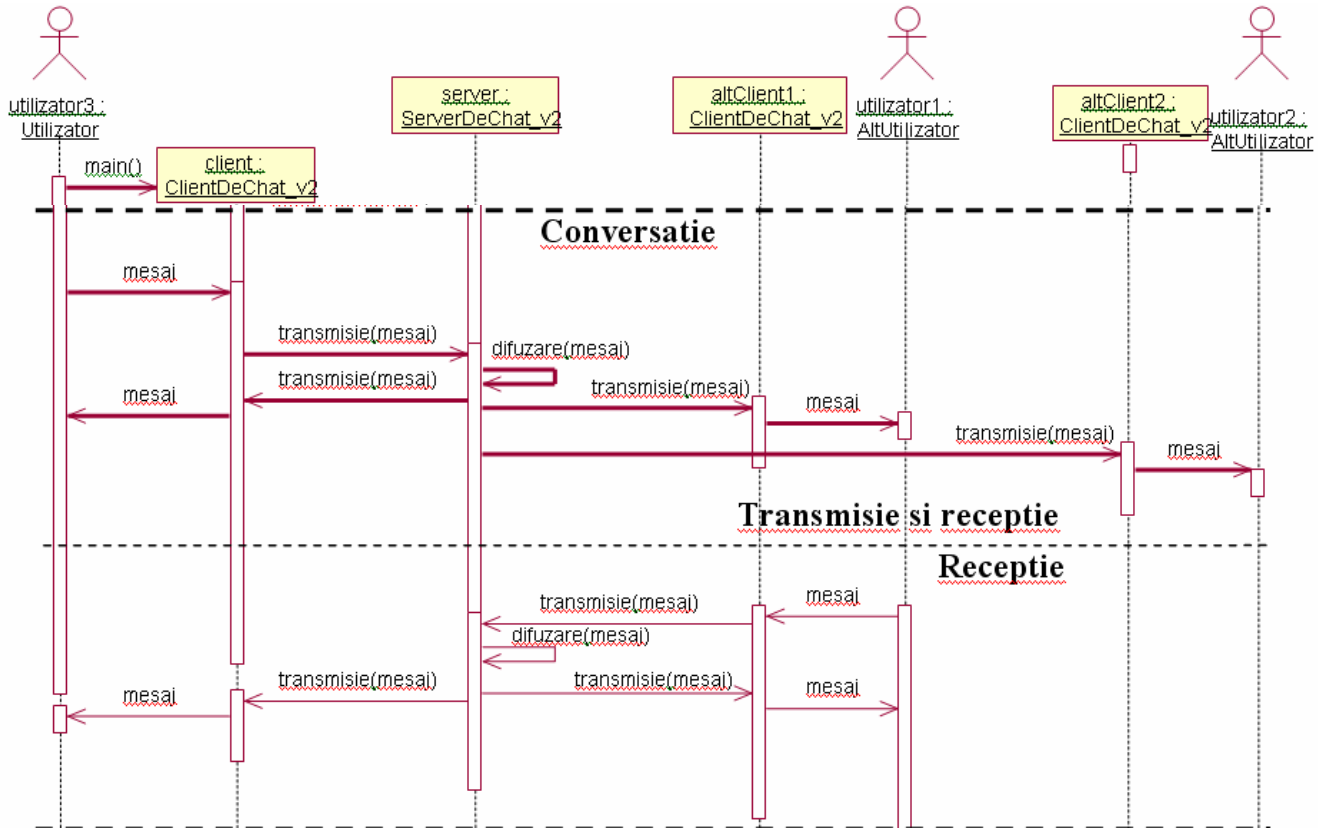
Varianta de nivel inalt a diagramei de secventa pentru **scenariul conectarii unui nou client.**



Varianta de nivel inalt a diagramei de secventa pentru **scenariul deconectarii unui client.**



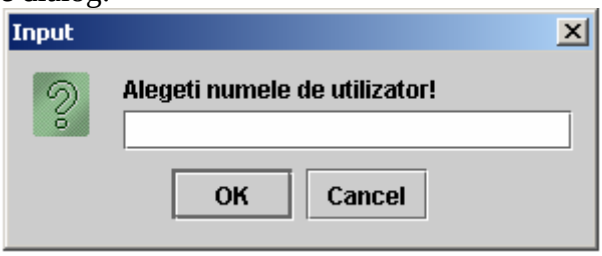
Varianta de nivel inalt a diagramei de secventa pentru **scenariul conversatiei intre clienti.**



P.2.4. Actualizarea proiectarii sistemului

Actualizarea proiectarii OO presupune:
- actualizarea proiectarii interfetei grafice cu utilizatorul
- actualizarea diagramelor UML de secventa si de colaborare
- actualizarea diagramelor UML de clase
- actualizarea diagramelor UML de stari
- actualizarea diagramelor UML de activitati

Interfata grafica prezentata initial unui nou utilizator, pentru alegerea numelui in vederea conectarii, este o fereastra de dialog:



In cazul in care utilizatorul propune un nume care exista deja in lista serverului, acesta respinge inregistrarea iar **clientul prezinta urmatoarea fereastra de dialog**:

Presupunem urmatorul scenariu:

- primul utilizator isi alege numele **isw2004**,

- al doilea utilizator isi alege numele **swrtc2004** (moment in care ambii utilizatori sunt informati, unul de existenta celuilalt),

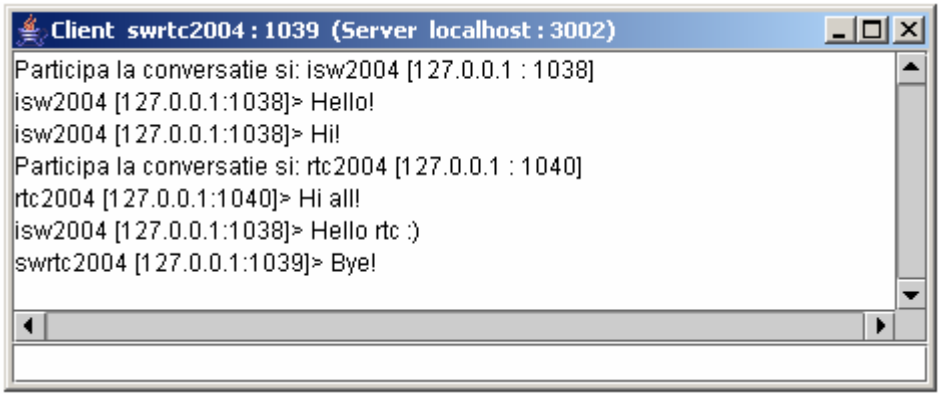
- cei doi utilizatori schimba cateva mesaje,

- se conecteaza al treilea utilizator, care isi alege numele **rtc2004** (moment in care primii doi utilizatori sunt informati de existenta celui de-al treilea, iar acesta de existenta celorlalti doi),

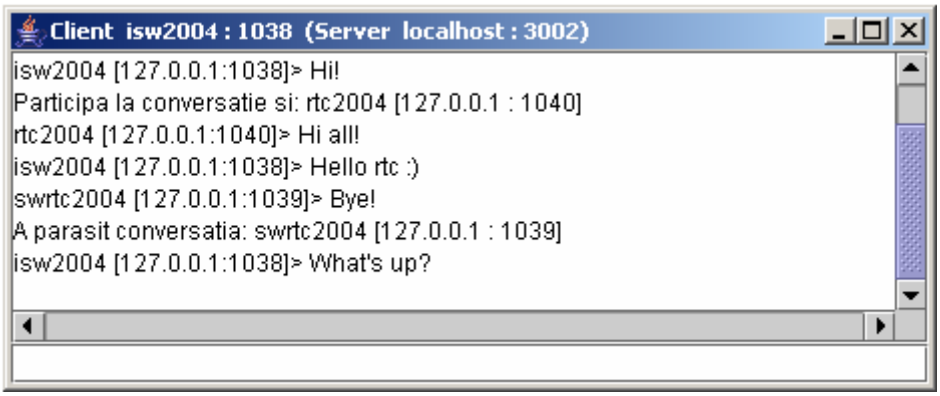
- cei trei utilizatori schimba cateva mesaje,

- utilizatorul cu numele **swrtc2004** paraseste conversatia (moment in care ceilalti doi utilizatori sunt informati).

Continutul interfetei grafice pentru cei trei utilizatori va fi urmatorul:
- pentru utilizatorul cu numele **swrtc2004**:



- pentru utilizatorul cu numele **isw2004**:



- pentru utilizatorul cu numele **rtc2004**:

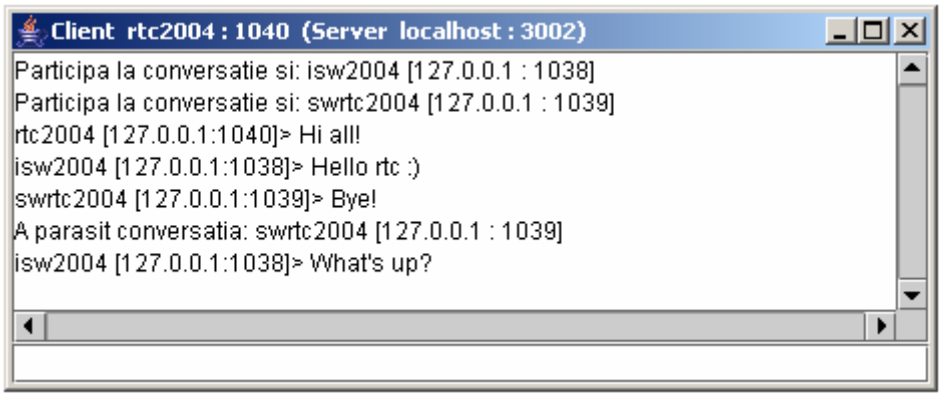


Diagrama de clase actualizata a **clientului** este urmatoarea:

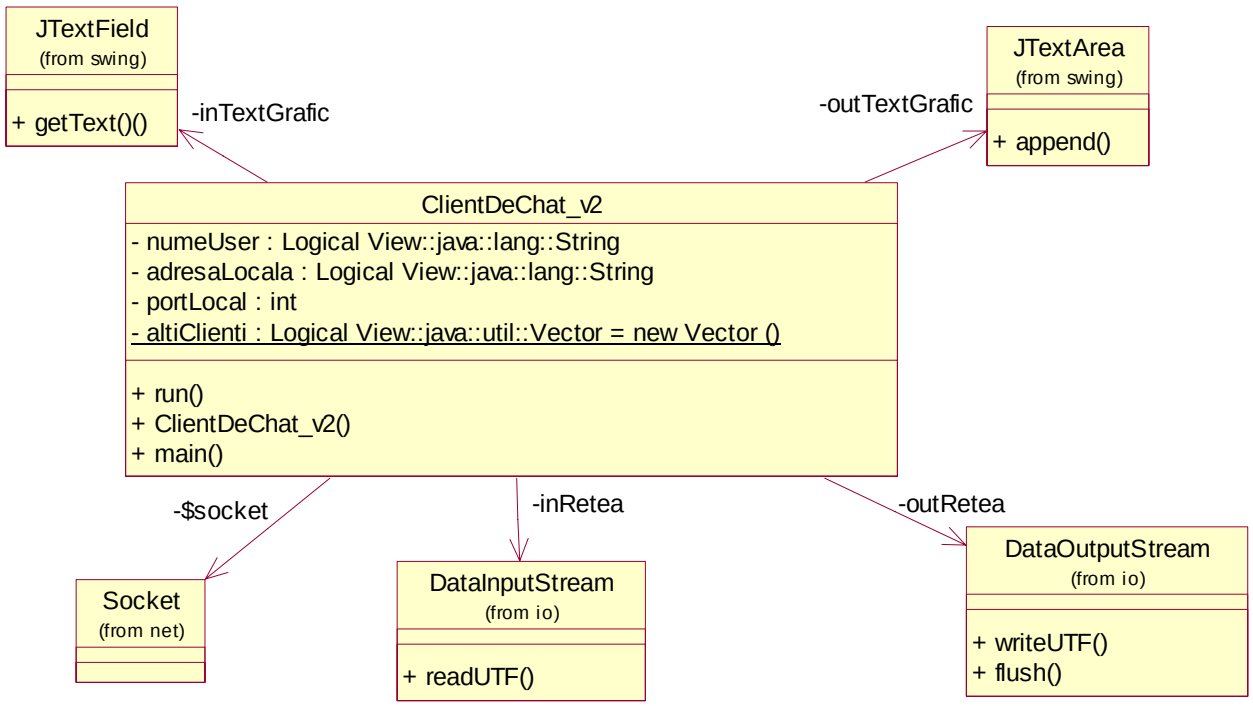
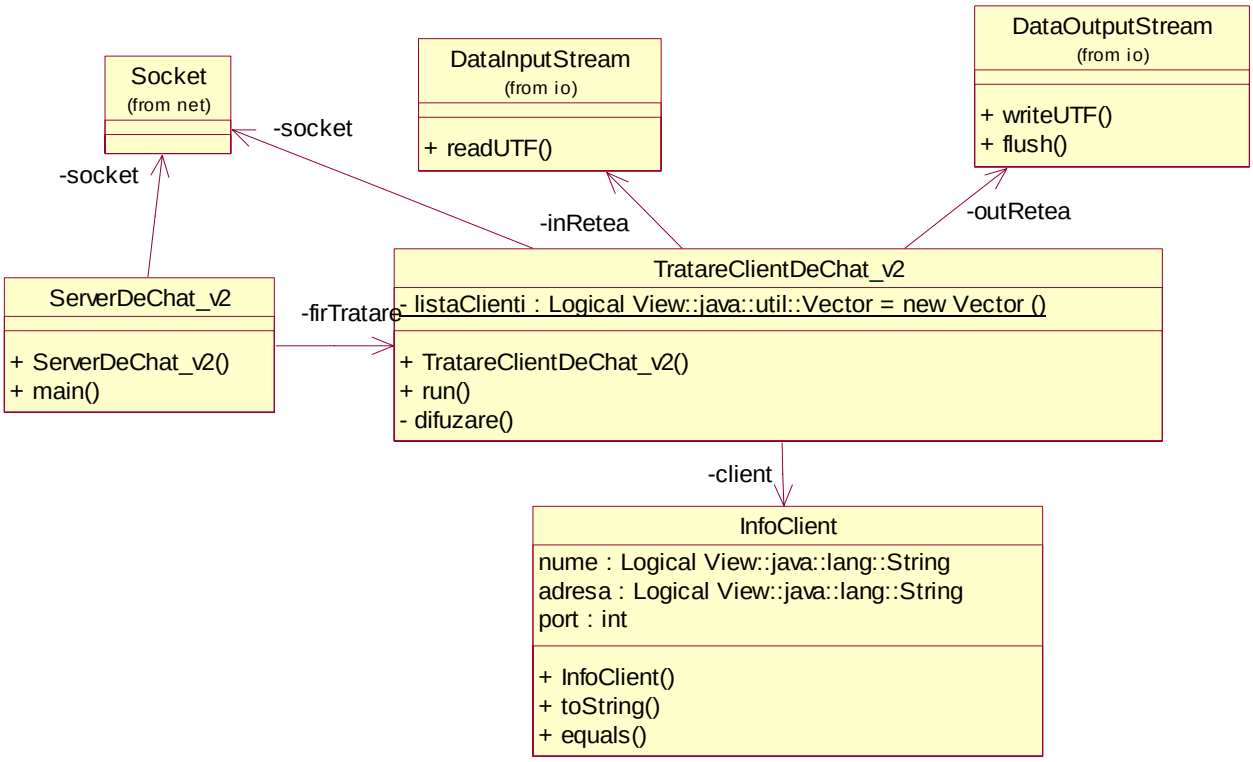


Diagrama de clase actualizata a **serverului** este urmatoarea:



P.2.5. Actualizarea implementarii OO a sistemului

Codul sursa Java pentru **clientul de chat** (si fisierul pentru compilarea si lansarea clientului)

```
1  //Source file: ClientDeChat_v2.java
2
3  import java.net.*;
4  import java.io.*;
5  import java.util.*;
6  import java.awt.*;
7  import java.awt.event.*;
8  import javax.swing.*;
9  import java.util.Properties;
10
11 /**
12  * Client de chat simplu - aplicatie de sine statatoare
13  * Aplicatie grafica Swing (extinde JFrame)
14  * care poate lansa in executie un fir nou (implementeaza Runnable)
15  */
16 public class ClientDeChat_v2 extends JFrame implements Runnable {
17   private String numeUser;
18   private String adresaLocala;
19   private int portLocal;
20
21   /**
22    * Vector de obiecte cu informatii privind ceilalti clienti
23    */
24   private static Vector altiClienti = new Vector ();
25
26   /**
27    * Flux de intrare dinspre retea
28    */
29   private DataInputStream inRetea;
30
31   /**
32    * Zona de text (configurata ca non-editabila)
33    */
34   private JTextArea outTextGrafic;
```

```
35
36    /**
37     * Intrare de text (editabila)
38     */
39    private JTextField inTextGrafic;
40
41    /**
42     * Fir de executie
43     */
44    private Thread firReceptie;
45
46    /**
47     * Socket flux (TCP)
48     */
49    private static Socket socket;
50
51    /**
52     * Flux de iesire catre retea
53     */
54    private DataOutputStream outRetea;
55    private InetAddress iaLocala;
56
57    JScrollBar baraDefilareVerticala;
58
59    /**
60     * Initializeaza obiectul de tip ClientDeChat_v1
61     * @param title      Titlul ferestrei
62     * @param inRetea    Flux de intrare dinspre retea
63     * @param outRetea   Flux de iesire catre retea
64     */
65    public ClientDeChat_v2(String title, InputStream inRetea,
66                                         OutputStream outRetea) {
67      // Stabilire titlu fereastra (JFrame)
68      super();
69      // ---------------------------------------------------------------
70      // Trimiterea catre server a informatiilor privind clientul
71      // ---------------------------------------------------------------
72
73      this.inRetea = new DataInputStream (new BufferedInputStream (inRetea));
74      this.outRetea = new DataOutputStream (new BufferedOutputStream (outRetea));
75
76      // ------------------------------------------------------------------------
77      // Afisarea in consola si trimiterea catre server a info. privind clientul
78      // ------------------------------------------------------------------------
79
80      // Obtinerea numarului de port local
81      portLocal = this.socket.getLocalPort();
82
83      try {
84        // Obtinerea adresei locale ca obiect InetAddress
85        iaLocala = InetAddress.getLocalHost();
86
87        // Obtinerea formei String a adresei locale
88        adresaLocala = iaLocala.getHostAddress();
89
90        System.out.println("\n Clientul are adresa " + adresaLocala +
91                           " si portul " + portLocal);
92
93        // Scrierea adresei IP pe fluxul de iesire spre retea
94        this.outRetea.writeUTF(adresaLocala);
95
96        // Scrierea portului TCP pe fluxul de iesire spre retea
97        this.outRetea.writeInt(portLocal);
98
99        // ---------------------------------------------------------
00        // Trimiterea (si eventual modificarea) numelui utilizatorului
01        // ---------------------------------------------------------
02
03        boolean existaDeja;
04
05        String mesajDialog = "Alegeti numele de utilizator!";
06
07
```

```
08
09        do {
10          // Alegerea numelui de utilizator
11          numeUser = JOptionPane.showInputDialog(mesajDialog);
12
13          // Scrierea numelui utilizatorului pe fluxul de iesire spre retea
14          this.outRetea.writeUTF(numeUser);
15          // Fortarea trimiterii (fortarea golirii bufferului)
16          this.outRetea.flush ();
17
18          // Citirea rezultatului din fluxul de intrare dinspre retea
19          String rezultat = this.inRetea.readUTF();
20
21          existaDeja = rezultat.equals("Numele a fost deja alocat");
22
23          mesajDialog = "Numele ales exista deja. Alegeti alt nume!";
24
25        } while (existaDeja);
26
27        System.out.println("\n Clientul are numele " + numeUser);
28      }
29
30      // ---------------------------------------------------------
31      // Tratarea erorilor de conexiune
32      // ---------------------------------------------------------
33      catch (IOException ex) {
34        // Afisarea exceptiei
35        ex.printStackTrace();
36      }
37
38      // ---------------------------------------------------------
39      // Initializari grafice
40      // ---------------------------------------------------------
41      this.setTitle("Client  " + numeUser + " : " + title);
42
43      Container containerCurent = this.getContentPane();
44
45      containerCurent.setLayout(new BorderLayout());
46
47      // Zona de text non-editabila de iesire (cu posibilitati de defilare)
48      outTextGrafic = new JTextArea(8, 40);
49      JScrollPane panouDefilabil = new JScrollPane(outTextGrafic,
50                                   JScrollPane.VERTICAL_SCROLLBAR_ALWAYS,
51                                   JScrollPane.HORIZONTAL_SCROLLBAR_ALWAYS);
52      baraDefilareVerticala = panouDefilabil.getVerticalScrollBar();
53      containerCurent.add("Center", panouDefilabil);
54      outTextGrafic.setEditable (false);
55
56      // Camp de text editabil de intrare
57      inTextGrafic = new JTextField(40);
58      containerCurent.add("South", inTextGrafic);
59
60      System.out.println("\n Configurata interfata grafica \n");
61
62      // ---------------------------------------------------------
63      // Tratarea evenimentelor interfetei grafice (intrarea de text)
64      // ---------------------------------------------------------
65
66      // Variabila locala finala (folosita in clasa interna anonima de tip
67      // ActionListener)
68      final DataOutputStream outR = this.outRetea;
69
70      // Crearea unui "ascultator" de "evenimente actionare"
71      ActionListener ascultatorInText = new ActionListener() {
72
73        // Tratarea actionarii intrarii de text (apasarea tastei "Enter")
74        public void actionPerformed(ActionEvent ev) {
75
76          // Citire mesajului din intrarea de text
77          String intrare = inTextGrafic.getText();
78
79
80
```

```
81
82          try {
83            // Scrierea mesajului pe fluxul de iesire spre retea
84            outR.writeUTF(numeUser + " [" + adresaLocala + ":" + portLocal +
85                          "]> " + intrare);
86            // Fortarea trimiterii mesajului (fortarea golirii bufferului)
87            outR.flush ();
88          }
89          // In cazul unei erori legata de conexiune
90          catch (IOException ex) {
91            // Afisarea exceptiei
92            ex.printStackTrace();
93            // Inchiderea firului de executie care efectueaza receptia
94            firReceptie = null;
95          }
96          // Pregatirea intrarii de text pentru noul mesaj (golirea intrarii)
97          inTextGrafic.setText ("");
98        }
99      };
00
01      // Inregistrarea "ascultatorului" de "evenimente actionare" la
02      // "obiectul sursa" intrare text
03      inTextGrafic.addActionListener(ascultatorInText);
04
05      // ---------------------------------------------------------------
06      // Tratarea evenimentelor interfetei grafice (inchiderea ferestrei)
07      // ---------------------------------------------------------------
08      // Crearea unui "adaptor pentru ascultator" de "evenimente fereastra"
09      WindowAdapter ascultatorInchidere = new WindowAdapter() {
10
11        // Tratarea inchiderii ferestrei curente
12        public void windowClosing(WindowEvent ev) {
13          // Daca mai exista firul de executie de receptie
14          if (firReceptie != null) {
15            // Inchiderea firului de receptie
16            firReceptie = null;
17          }
18          // Terminarea programului
19          System.exit(0);
20        }
21      };
22      // Inregistrarea "ascultatorului" de "evenimente fereastra" la "sursa"
23      // (fereastra curenta)
24      this.addWindowListener(ascultatorInchidere);
25
26      // ---------------------------------------------------------
27      // Lansarea interfetei grafice
28      // ---------------------------------------------------------
29
30      // Impachetarea (compactarea) componentelor in container
31      pack();
32      // Fereastra devine vizibila - echivalent cu frame.setVisible(true)
33      show();
34      // Cerere focus pe intrarea de text din fereastra curenta
35      inTextGrafic.requestFocus();
36
37      // ---------------------------------------------------------
38      // Lansarea firului de executie de receptie
39      // ---------------------------------------------------------
40
41      // Fir de executie pentru receptia mesajelor de la server
42      firReceptie = new Thread (this);
43      // Lansarea firului de executie - se executa run()
44      firReceptie.start ();
45    }
46
47
48
49
50
51
52
53
```

```
54    /**
55     * Metoda principala a firului care receptioneaza mesaje de la server
56     */
57    public void run() {
58
59      // ------------------------------------------------------------
60      // Tratarea mesajelor serverului (citirea si interpretarea lor)
61      // ------------------------------------------------------------
62      try {
63        while (true) {
64          // Citirea mesajului din fluxul de intrare dinspre server
65          String line = inRetea.readUTF ();
66
67          // ------------------------------------------------------------
68          // Inregistrarea unui client nou sau preexistent
69          // ------------------------------------------------------------
70
71          if (line.equals("Client nou")) {
72            // Citirea mesajului din fluxul de intrare dinspre server
73            String numeClient = inRetea.readUTF ();
74            // Citirea mesajului din fluxul de intrare dinspre server
75            String adresaClient = inRetea.readUTF ();
76            // Citirea mesajului din fluxul de intrare dinspre server
77            int portClient = Integer.parseInt(inRetea.readUTF());
78
79            // Incapsularea informatiilor privind clientul
80            InfoClient clientNou = new
81                         InfoClient(numeClient, adresaClient, portClient);
82
83            // Inregistrarea clientului curent in lista (Vector)
84            altiClienti.addElement (clientNou);
85
86            System.out.println("Client nou: " + numeClient +
87                               " [" + adresaClient +
88                               " : " + portClient + "]");
89
90            System.out.println("Lista clientilor:\n" + altiClienti.toString());
91
92            // Anuntarea in iesirea de text a noului client
93            outTextGrafic.append ("Participa la conversatie si: " + numeClient +
94                               " [" + adresaClient +
95                               " : " + portClient + "] \n");
96          }
97
98          // ------------------------------------------------------------
99          // Eliminarea inregistrarii unui client care a parasit conversatia
00          // ------------------------------------------------------------
01
02          else if (line.equals("Client eliminat")) {
03            // Citirea mesajului din fluxul de intrare dinspre server
04            String numeClient = inRetea.readUTF ();
05            // Citirea mesajului din fluxul de intrare dinspre server
06            String adresaClient = inRetea.readUTF ();
07            // Citirea mesajului din fluxul de intrare dinspre server
08            int portClient = Integer.parseInt(inRetea.readUTF());
09
10            // Incapsularea informatiilor privind clientul
11            InfoClient clientEliminat = new
12                         InfoClient(numeClient, adresaClient, portClient);
13
14            altiClienti.removeElement (clientEliminat);
15
16            System.out.println("Client eliminat: " + numeClient +
17                               " [" + adresaClient +
18                               " : " + portClient + "]");
19
20            System.out.println("Lista clientilor:\n" + altiClienti.toString());
21
22            // Anuntarea in iesirea de text a eliminarii clientului
23            outTextGrafic.append ("A parasit conversatia: " + numeClient +
24                               " [" + adresaClient +
25                               " : " + portClient + "] \n");
26          }
```

```
27
28          // ------------------------------------------------------------
29          // Afisarea mesajului (propriu-zis) receptionat
30          // ------------------------------------------------------------
31
32          else {
33            // Adaugare textului primit in iesirea de text
34            outTextGrafic.append (line + "\n");
35          }
36
37          // Solutie pentru a avea afisate intotdeauna ultimele
38          // mesaje primite (exista o solutie alternativa???)
39          int zonaVizibila = baraDefilareVerticala.getVisibleAmount();
40          int valoareaMaxima = baraDefilareVerticala.getMaximum();
41
42          baraDefilareVerticala.setValue(valoareaMaxima - zonaVizibila);
43          validate();
44          repaint();
45        }
46      }
47
48      // ------------------------------------------------------------
49      // Tratarea erorilor de conexiune
50      // ------------------------------------------------------------
51
52      catch (IOException ex) {
53        // Afisarea exceptiei
54        ex.printStackTrace ();
55      }
56
57      // ------------------------------------------------------------
58      // Curatenie finala
59      // ------------------------------------------------------------
60
61      finally {
62        // Inchiderea firului de receptie curent
63        firReceptie = null;
64        // Ascunderea intrarii de text
65        inTextGrafic.setVisible(false);
66        // Reasezarea interfetei grafice
67        validate ();
68        try {
69          // Inchiderea fluxului de iesire spre retea
70          outRetea.close ();
71          }
72        // In cazul unei erori legata de conexiune
73        catch (IOException ex) {
74          // Afisarea exceptiei
75          ex.printStackTrace ();
76        }
77      }
78    }
79
80
81    /**
82     * Metoda principala - creaza socketul, fluxurile si lanseaza clientul
83     * @param args[]
84     * @param args
85     * @throws java.io.IOException
86     */
87    public static void main(java.lang.String[] args) throws IOException {
88
89      // Adresa serverului
90      String adresaServer =
91               JOptionPane.showInputDialog("Introduceti adresa serverului");
92
93      // Portul serverului
94      String numarPort = JOptionPane.showInputDialog(
95                             "Introduceti numarul de port al serverului");
96
97      int portServer = Integer.parseInt(numarPort);
98
99
```

```
00      // Crearea socketului catre server
01      socket = new Socket (adresaServer, portServer);
02
03      System.out.println ("\n Client TCP lansat catre serverul  [" +
04                      socket.getInetAddress() + ":" + socket.getPort() + "]");
05
06      // Crearea fluxurilor, crearea si lansarea clientului grafic
07      new ClientDeChat_v2 (socket.getLocalPort() +
08                      "  (Server  " + adresaServer + " : " + numarPort + ")",
09                         socket.getInputStream (), socket.getOutputStream ());
10
11    }
12  }
13
14  class InfoClient {
15    String nume;
16    String adresa;
17    int port;
18    Thread fir;
19
20    /**
21     * @param nume
22     * @param adresa
23     * @param port
24     * @param fir
25     */
26    public InfoClient(String nume, String adresa, int port, Thread fir) {
27      this.nume = nume;
28      this.adresa = adresa;
29      this.port = port;
30      this.fir = fir;
31    }
32
33    /**
34     * @param nume
35     * @param adresa
36     * @param port
37     * @param fir
38     */
39    public InfoClient(String nume, String adresa, int port) {
40      this.nume = nume;
41      this.adresa = adresa;
42      this.port = port;
43      this.fir = null;
44    }
45
46    public void setFir(Thread fir) {
47      this.fir = fir;
48    }
49
50    /**
51     * @return java.lang.String
52     */
53    public String toString() {
54      return "\t" + nume + "/" + adresa + ":" + port + "\n";
55    }
56
57    /**
58     * @return boolean
59     */
60    public boolean equals(Object obj) {
61      if ((obj != null) && (obj instanceof InfoClient)) {
62        InfoClient celalaltClient = (InfoClient)obj;
63        return ((this.nume.equals(celalaltClient.nume)) &&
64                (this.adresa.equals(celalaltClient.adresa)) &&
65                (this.port == celalaltClient.port));
66      }
67      return false;
68    }
69  }
```

Codul sursa Java pentru **serverul de chat** (si fisierul pentru compilarea si lansarea serverului)

```
1  //Source file: ServerDeChat_v2.java
2
3  /**
4   * Server chat simplu - componenta server pentru noi conexiuni
5   */
6  import java.net.*;
7  import java.io.*;
8  import java.util.*;
9
10 public class ServerDeChat_v2 {
11   private TratareClientDeChat_v2 firTratare;
12   private Socket socket;
13
14   public ServerDeChat_v2(int port) throws IOException {
15
16     // Server pentru asteptarea cererilor de conectare
17     ServerSocket server = new ServerSocket (port);
18     System.out.println ("Server TCP lansat pe port " + port  + "...");
19
20     // Bucla infinita
21     while (true) {
22
23       // Asteptarea cererilor de conectare si returnarea unui nou socket
24       Socket client = server.accept ();
25       System.out.println ("Acceptata conexiunea de la: [" +
26                           client.getInetAddress().getHostAddress() + "(" +
27                           client.getInetAddress().getHostName() + ")" +
28                           ":" + client.getPort() + "]...");
29       System.out.println ("pe portul local: " + client.getLocalPort());
30
31       // Crearea unui fir de executie pentru tratare client nou
32       firTratare = new TratareClientDeChat_v2 (client);
33
34       // Lansarea firului de executie - se va executa: firTratare.run()
35       firTratare.start ();
36      }
37   }
38
39   public static void main(java.lang.String[] args) throws IOException {
40     if (args.length != 1)
41       throw new RuntimeException ("Sintaxa: ServerDeChat_v2 <numarPort>");
42
43     new ServerDeChat_v2 (Integer.parseInt (args[0]));
44   }
45 }
```

Codul sursa Java pentru **firul de tratare a clientului** de chat.

```
1  //Source file: TratareClientDeChat_v2.java
2
3  /**
4   * Server chat simplu - componenta de tratare a unei conexiuni
5   */
6  import java.net.*;
7  import java.io.*;
8  import java.util.*;
9  import java.util.Vector;
10
11 public class TratareClientDeChat_v2 extends Thread {
12
13   /**
14    * Vector de referinte la obiecte care trateaza clienti (ptr. inregistrare)
15    */
16   private static Vector listaClienti = new Vector ();
17
18   /**
19    * Socket flux (TCP)
20    */
21   private Socket socket;
```

```
22
23    /**
24     * Flux de intrare dinspre retea
25     */
26    private DataInputStream inRetea;
27
28    /**
29     * Flux de iesire catre retea
30     */
31    private DataOutputStream outRetea;
32    private InfoClient client;
33
34
35    /**
36     * Initializeaza obiectul (firul) care trateaza un nou client
37     * @param socket
38     * @throws java.io.IOException
39     */
40    public TratareClientDeChat_v2(Socket socket) throws IOException {
41      this.socket = socket;
42      inRetea = new DataInputStream (new
43                           BufferedInputStream (socket.getInputStream ()));
44      outRetea = new DataOutputStream (new
45                           BufferedOutputStream (socket.getOutputStream ()));
46    }
47
48    /**
49     * Metoda principala a firului de executie.
50     * Primeste mesajele si apeleaza difuzarea lor.
51     */
52    public void run() {
53
54      try {
55
56        // ------------------------------------------------------------
57        // Obtinerea informatiilor de la client
58        // ------------------------------------------------------------
59
60        // Citirea adresei clientului din fluxul de intrare de la client
61        String adresaClient = inRetea.readUTF();
62
63        // Citirea portului clientului din fluxul de intrare de la client
64        int portClient = inRetea.readInt();
65
66        // ------------------------------------------------------------------------
67        // Obtinerea si verificarea (eventual modificarea) numelui utilizatorului
68        // ------------------------------------------------------------------------
69        // Numele clientului
70        String numeClient;
71
72        // Rezultatul verificarii existentei numelui in lista
73        boolean existaDeja;
74
75        // Receptia numelui si eventuala cerere de modificare
76        do {
77          // Citirea numelui clientului din fluxul de intrare de la client
78          numeClient = inRetea.readUTF();
79
80          existaDeja = false;
81
82          // Enumerare creata pornind de la lista clientilor
83          Enumeration enum = listaClienti.elements ();
84
85          // Cat timp mai sunt elemente in enumerare
86          while (enum.hasMoreElements ()) {
87
88            // Protectie la acces concurent la Vectorul clientilor
89            synchronized (listaClienti) {
90
91              // Obtinerea informatiilor privind clientul curent tratat
92              InfoClient infoClient = (InfoClient) enum.nextElement();
93
94              if (numeClient.equals(infoClient.nume)) {
```

```
95                existaDeja = true;
96                break;
97              }
98            }
99          }
00          if (existaDeja) {
01            // Scrierea mesajului in fluxul de iesire al firului curent
02            outRetea.writeUTF("Numele a fost deja alocat");
03          }
04          else {
05            // Scrierea mesajului in fluxul de iesire al firului curent
06            outRetea.writeUTF("Client inregistrat");
07          }
08          // Fortarea trimiterii
09          outRetea.flush();
10
11        } while (existaDeja);
12
13        // ------------------------------------------------------------------
14        // Trimiterea listei clientilor deja existenti (catre clientul tratat)
15        // ------------------------------------------------------------------
16
17        // Enumerare creata pornind de la lista clientilor
18        Enumeration enum = listaClienti.elements ();
19
20        // Cat timp mai sunt elemente in enumerare
21        while (enum.hasMoreElements ()) {
22
23          // Protectie la acces concurent la Vectorul clientilor
24          synchronized (listaClienti) {
25
26            // Obtinerea informatiilor privind clientul curent tratat
27            InfoClient infoClient = (InfoClient) enum.nextElement();
28
29            // Scrierea mesajului in fluxul de iesire al firului curent
30            outRetea.writeUTF("Client nou");
31            outRetea.writeUTF(infoClient.nume);
32            outRetea.writeUTF(infoClient.adresa);
33            outRetea.writeUTF(new Integer(infoClient.port).toString());
34          }
35        }
36
37        // Fortarea trimiterii
38        outRetea.flush();
39
40        // ------------------------------------------------------------------
41        // Anuntarea existentei clientului curent (catre ceilalti clienti)
42        // ------------------------------------------------------------------
43        System.out.println("Difuzare informatii client nou ");
44        difuzare("Client nou");
45        difuzare(numeClient);
46        difuzare(adresaClient);
47        difuzare(new Integer(portClient).toString());
48
49        // ------------------------------------------------------------------
50        // Inregistrarea informatiilor clientului tratat
51        // ------------------------------------------------------------------
52
53        // Inregistrarea informatiilor clientului curent
54        client = new InfoClient(numeClient, adresaClient, portClient, this);
55
56        // Inregistrarea clientului curent in lista (de tip Vector)
57        listaClienti.addElement (client);
58
59        // ------------------------------------------------------------------
60        // Afisarea in consola a informatiilor clientului curent tratat
61        // ------------------------------------------------------------------
62
63        System.out.println("\nNoul client [" + numeClient +
64                           "] are adresa [" + adresaClient +
65                           "] si portul [" + portClient +"]");
66        System.out.println("Lista clientilor actualizata:\n" +
67                                              listaClienti.toString()+"\n");
```

```
68
69        // ------------------------------------------------------------
70        // Tratarea mesajelor clientului (citirea si difuzarea mesajelor)
71        // ------------------------------------------------------------
72
73        // Bucla infinita (repetare pana la inchiderea firului curent)
74        while (true) {
75
76          // Citirea mesajului din fluxul de intrare de la client
77          String mesaj = inRetea.readUTF ();
78
79          System.out.println("Difuzare mesaj: " + mesaj);
80
81          // Difuzarea catre toti clientii curent inregistrati
82          difuzare (mesaj);
83
84          System.out.println();
85        }
86      }
87
88      // ------------------------------------------------------------
89      // Tratarea erorilor de conexiune
90      // ------------------------------------------------------------
91
92      catch (IOException ex) {
93        // Afisare exceptie
94        ex.printStackTrace ();
95      }
96
97      // ------------------------------------------------------------
98      // Curatenie finala
99      // ------------------------------------------------------------
00
01      finally {
02        // Eliminarea clientului curent din lista (de tip Vector)
03        listaClienti.removeElement (client);
04
05        System.out.println("\nClient eliminat...");
06        System.out.println("Lista clientilor actualizata:\n" +
07                                          listaClienti.toString()+"\n");
08
09        // ---------------------------------------------------------------
10        // Anuntarea eliminarii clientului curent (catre ceilalti clienti)
11        // ---------------------------------------------------------------
12        System.out.println("Difuzare informatii client eliminat ");
13        difuzare("Client eliminat");
14        difuzare(client.nume);
15        difuzare(client.adresa);
16        difuzare(new Integer(client.port).toString());
17
18        try {
19          // Inchiderea socketului
20          socket.close ();
21        }
22        // In cazul unei erori legata de conexiune
23        catch (IOException ex) {
24          // Afisarea exceptiei
25          ex.printStackTrace ();
26        }
27      }
28    }
29
30
31
32
33
34
35
36
37
38
39
40
```

```
41    /**
42     * Difuzeaza mesajul primit catre clienti
43     * @param mesaj
44     */
45    private static void difuzare(String mesaj) {
46
47      // Enumerare creata pornind de la lista clientilor
48      Enumeration enum = listaClienti.elements ();
49
50      // Cat timp mai sunt elemente in enumerare
51      while (enum.hasMoreElements ()) {
52
53        // Referinta catre firul curent initializata cu null
54        TratareClientDeChat_v2 firDestinatie = null;
55
56        // Protectie la acces concurent la Vectorul clientilor
57        synchronized (listaClienti) {
58
59          // Obtinerea informatiilor privind clientul curent
60          InfoClient infoClient = (InfoClient) enum.nextElement();
61
62          System.out.print(" - catre: " + infoClient.toString());
63
64          // Obtinerea referintei catre firul curent
65          firDestinatie = (TratareClientDeChat_v2) infoClient.fir;
66
67          // Daca referinta e valida
68          if (firDestinatie  != null) {
69
70            try {
71             // Protectie la acces concurent la fluxul de iesire
72             synchronized (firDestinatie.outRetea) {
73
74                // Scrierea mesajului in fluxul de iesire al firului curent
75                firDestinatie.outRetea.writeUTF (mesaj);
76              }
77
78              // Fortarea trimiterii mesajului
79              firDestinatie.outRetea.flush ();
80            }
81
82            // In cazul unei erori legata de conexiune
83            catch (IOException ex) {
84              // Inchiderea firului curent
85              listaClienti.remove(infoClient);
86              firDestinatie = null;
87            }
88          }
89        }
90      }
91    }
92  }
93
94
95  class InfoClient {
96    String nume;
97    String adresa;
98    int port;
99    Thread fir;
00
01    /**
02     * @param nume
03     * @param adresa
04     * @param port
05     * @param fir
06     */
07    public InfoClient(String nume, String adresa, int port, Thread fir) {
08      this.nume = nume;
09      this.adresa = adresa;
10      this.port = port;
11      this.fir = fir;
12    }
13
```

```
14    /**
15     * @return java.lang.String
16     */
17    public String toString() {
18      return fir.toString() + "(" + nume + "/" + adresa + ":" + port + ")\n";
19    }
20
21    /**
22     * @return boolean
23     */
24    public boolean equals(Object obj) {
25      if ((obj != null) && (obj instanceof InfoClient)) {
26        InfoClient celalaltClient = (InfoClient)obj;
27        return ((this.nume.equals(celalaltClient.nume)) &&
28                (this.adresa.equals(celalaltClient.adresa)) &&
29                (this.port == celalaltClient.port));
30      }
31      return false;
32    }
33  }
```