Catedra de Telecomunicatii

13/01/2008

Programare Orientata spre Obiecte (POO)

Laborator 6

Fire de executie (*threads*) Java. Aplicatii multifilare bazate pe *socket*-uri TCP si interfete grafice Swing

6.1. Descrierea laboratorului

In aceasta lucrare de laborator vor fi acoperite urmatoarele probleme:

- **Utilizarea firelor de executie (clasa `Threads`)** – Introducere in fire de executie – **link extern**
- **Aplicatii client-server cu interfata grafica Swing bazate pe socket-uri TCP**
- **Precizari privind mini-proiectul pentru examenul final**

6.2. Utilizarea firelor de executie (threads) in Java

6.2.1. Modalitati de a crea un fir de executie

Pentru a crea un nou fir de executie (detalii privind firele de executie - *threads* - in Java) **exista doua modalitati**.

1. Se poate declara o clasa ca subclasa a clasei **`Thread`**, subclasa care **trebuie sa rescrie codul (*override*) metodei `run()` a clasei `Thread`** (care nu contine nici un cod), noul fir de executie fiind creat prin alocarea si lansarea unei instante a subclasei.

Formatul pentru crearea unei subclase care extinde clasa `Thread` si ii reimplementeaza metoda `run()` este urmatorul:

```
class FirT extends Thread {
   public void run() {
                          // codul firului de executie
   }
}
```

Formatul pentru crearea unei instante a subclasei este urmatorul:

```
FirT fir = new FirT();
```

2. Se poate declara o clasa care implementeaza interfata **`Runnable`**, interfata care contine doar declaratia unei metode **`run()`** (declaratie similara celei din clasa `Thread`, clasa `Thread` implementand interfata `Runnable`), noul fir fiind obtinut prin **crearea unei instante a noii clase**, **pasata constructorului** la **crearea unei instante a clasei `Thread`**, si **lansarea noii instante a clasei `Thread`**.

Formatul pentru crearea unei clase care implementeaza interfata `Runnable` si ii implementeaza metoda `run()` este urmatorul:

```
class FirR implements Runnable {
   public void run() {
                          // codul firului de executie
   }
}
```

Formatul pentru crearea unei instante a noii clase si apoi a unei instante a clasei `Thread` este urmatorul:

```
FirR r = new FirR();
Thread fir = new Thread(r);
```

In ambele cazuri formatul pentru lansarea noului fir de executie, este urmatorul:

```
fir.start();
```

Variante compacte pentru crearea si lansarea noilor fire de executie:

```
new FirT().start();                        // nu exista variabila de tip FirT
                                           // care sa refere explicit firul
```

```
FirR r = new FirR();
new Thread(r).start();                     // nu exista variabila de tip Thread
                                           // care sa refere explicit firul
```

```
Thread fir = new Thread(new FirR());       // nu exista variabila de tip FirR
fir.start();                               // care sa refere explicit firul
```

```
new Thread(new FirR()).start();            // nu exista variabila de tip Thread
                                           // si nici variabila de tip FirR
                                           // care sa refere explicit firul
```

6.2.2. Programe multifilare (*multi-threaded*)

Urmatorul program va fi folosit pentru a exeplifica **lucrul cu fire de executie**. Clasa **`FirSimplu`** extinde clasa **`Thread`**, iar **metoda principala lanseaza** metoda **`run`**`()` ca **nou fir de executie**.

```
 1  public class FirSimplu extends Thread { // obiectele din clasa curenta sunt thread-uri
 2    public FirSimplu(String str) {
 3        super(str);                       // invocarea constructorului Thread(String)
 4    }                                     // al superclasei Thread
 5    public void run() {                   // "metoda principala" a thread-ului curent
 6        for (int i = 0; i < 5; i++) {
 7            System.out.println(i + " " + getName()); // obtinerea numelui thread-ului
 8            try {
 9                sleep((long)(Math.random() * 1000)); // thread-ul "doarme" 0...1 secunda
10            } catch (InterruptedException e) {}
11        }
12        System.out.println("Gata! " + getName());    // obtinerea numelui thread-ului
13    }
14    public static void main (String[] args) {
15        new FirSimplu("Unu").start();         // "lansarea" thread-ului (apeleaza run())
16    }
17  }
```

Clasa **`DemoTreiFire`** **lanseaza trei fire** de executie de tip **`FirSimplu`** executate **concurent**.

```
1  public class DemoTreiFire {
2     public static void main (String[] args) {
3         new FirSimplu("Unu").start();     // "lansarea" primului thread
4         new FirSimplu("Doi").start();     // "lansarea" celui de-al doilea thread
5         new FirSimplu("Trei").start();    // "lansarea" celui de-al treilea thread
6     }
7  }
```

Firele au evolutii diferite, in functie de durata intarzierii introdusa in `FirSimplu` de linia:

```
sleep( (long) (Math.random() * 1000) );
```

In laborator:

1. **Inchideti proiectele anterioare** (**Ctrl+W**). **Creati un nou proiect** ***fire***
2. **Creati o noua clasa**, numita **FirSimplu**, folosind **codul dat mai sus**.
3. **Creati un obiect** tip **FirSimplu** (cu nume **`"Test"`**), **inspectati-i campurile** (atributele) **si metodele**.
4. **Tot in proiectul *fire* creati** clasa **DemoTreiFire** folosind **codul de mai sus**. **Compilati clasele.**
5. ***Right-click*** pe clasa, selectati si executati **main()**. Urmariti **efectul** in **Terminal Window**.

6.3. Aplicatii client-server bazate pe socket TCP si interfete grafice Swing

6.3.1. Sistem de comunicatie client-server bazat pe socket TCP si GUI Swing

Superclasa ElementeComuneClientServer contine elemente comune clientilor si serverelor si este mostenita si extinsa de clienti si servere:

```
 1   import java.net.*;
 2   import java.io.*;
 3   import javax.swing.JOptionPane;
 4
 5   public class ElementeComuneClientServer {
 6      protected ConexiuneRetea conexiune;
 7      protected Socket socketTCP;
 8      protected int portTCP;
 9
10      public ElementeComuneClientServer(String tip) throws IOException {
11          portTCP = Integer.parseInt(JOptionPane.showInputDialog(tip +
12                          ": introduceti numarul de port al serverului"));
13      }
14   }
```

Clasa utilitara ConexiuneRetea incapsuleaza tratarea conexiunilor TCP:

```
 1   import java.net.*;
 2   import java.io.*;
 3   import java.util.Scanner;
 4
 5   public class ConexiuneRetea {
 6      private Socket conexiune;
 7      private Scanner scannerTCP;
 8      private PrintStream printerTCP;
 9      public ConexiuneRetea(Socket conexiune) throws IOException {
10          this.conexiune = conexiune;
11          this.scannerTCP = new Scanner(conexiune.getInputStream());
12          this.printerTCP = new PrintStream(conexiune.getOutputStream());
13      }
14      public String nextLine() {
15          return this.scannerTCP.nextLine();
16      }
17      public int nextInt() {
18          return this.scannerTCP.nextInt();
19      }
20      public void printLine(String text) {
21          this.printerTCP.println(text);
22          this.printerTCP.flush();
23      }
24   }
```

Clasa ClientBidirectionalSwing mosteneste si extinde clasa ElementeComuneClientServer:

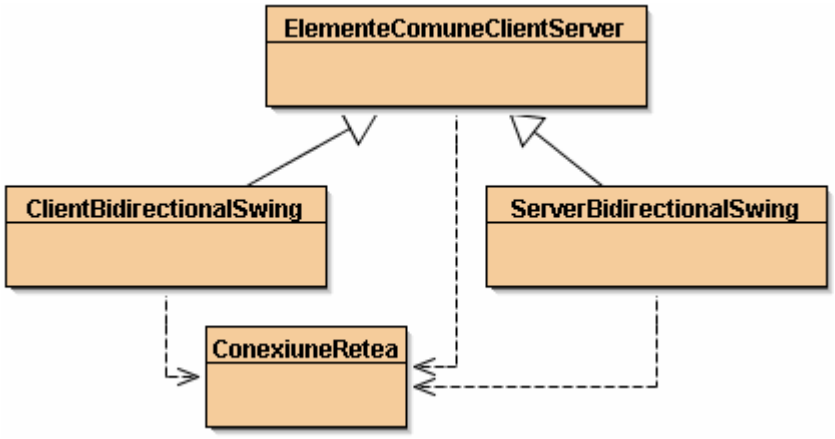
```
 1   import java.net.*;
 2   import java.io.*;
 3   import javax.swing.JOptionPane;
 4   import java.awt.*;
 5   import java.awt.event.*;
 6   import javax.swing.*;
 7
 8   public class ClientBidirectionalSwing extends ElementeComuneClientServer
 9                                         implements Runnable {
10      private InetAddress adresaIP;
11      private JFrame frame;
12      private JTextField inTextGrafic;   // Intrare - linie text grafica (JtextField)
13      private JTextArea outTextGrafic;   // Iesire - zona text grafica (JtextArea)
14      private JScrollBar vertical;
15      private String sirCitit;
16
17      public ClientBidirectionalSwing() throws IOException {
18          super("Client");
19          adresaIP = InetAddress.getByName(JOptionPane.showInputDialog(
20                          "Client: introduceti adresa serverului"));
21          socketTCP = new Socket(adresaIP, portTCP);           // Creare socket
22          conexiune = new ConexiuneRetea(socketTCP);
```

```
23
24          // Crearea obiectului cadru (frame), cu titlu specificat
25          JFrame frame = new JFrame("Client bidirectional Swing"); // Stabilire titlu
26          Container containerCurent = frame.getContentPane();
27          containerCurent.setLayout(new BorderLayout());
28
29          outTextGrafic = new JTextArea(5, 40);  // Zona text non-editabila de iesire
30          JScrollPane scrollPane = new JScrollPane(outTextGrafic,
31                      JScrollPane.VERTICAL_SCROLLBAR_ALWAYS,
32                      JScrollPane.HORIZONTAL_SCROLLBAR_ALWAYS);
33          vertical = scrollPane.getVerticalScrollBar();
34          containerCurent.add("Center", scrollPane);
35          outTextGrafic.setEditable(false);
36          outTextGrafic.append("Pentru oprire introduceti '.' si <Enter>\n\n");
37
38          inTextGrafic = new JTextField(40);      // Camp de text editabil de intrare
39          containerCurent.add("South", inTextGrafic);
40          inTextGrafic.requestFocus(); // Cerere focus pe intrarea text
41
42          // Inregistrare "ascultator" de "evenimente actionare" la "obiectul sursa"
43          inTextGrafic.addActionListener(new AscultatorInText());
44
45          // Iesire din program la inchiderea ferestrei
46          frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
47
48          frame.pack();      // Impachetarea (compactarea) componentelor in container
49          frame.setVisible(true);      // Fereastra devine vizibila
50      }
51
52      // Clasa interna "ascultator" de "evenimente actionare"
53      // implementeaza interfata ActionListener
54      class AscultatorInText implements ActionListener {
55
56        // Tratarea actionarii intrarii de text (introducerii unui "Enter")
57        public void actionPerformed(ActionEvent ev) {
58
59           sirCitit = inTextGrafic.getText();  // Citirea unei linii din intrarea text
60           inTextGrafic.setText("");           // Golirea intrarii text
61           conexiune.printLine(sirCitit);
62
63           outTextGrafic.append("S-a introdus: "+sirCitit+"\n"); // Scriere linie
64           // actualizeaza pozitia barei de defilare, varianta imbunatatita a
65           // vertical.setValue(vertical.getMaximum() - vertical.getVisibleAmount());
66           SwingUtilities.invokeLater(new Runnable()
67             { public void run() { vertical.setValue(vertical.getMaximum()); } });
68
69           if (sirCitit.equals(new String(".")))   System.exit(0); // Conditie oprire
70        }
71      }
72
73      public void run () {
74          String mesaj;
75          while (true) {
76            mesaj = conexiune.nextLine();
77
78            outTextGrafic.append("S-a primit:    " + mesaj + "\n");
79            // actualizeaza pozitia barei de defilare, varianta imbunatatita a
80            // vertical.setValue(vertical.getMaximum() - vertical.getVisibleAmount());
81            SwingUtilities.invokeLater(new Runnable()
82              { public void run() { vertical.setValue(vertical.getMaximum()); } });
83
84            if (mesaj.equals(".")) break;    // Testarea conditiei de oprire
85          }
86          try {  socketTCP.close();        // Inchiderea socketului si a fluxurilor
87          } catch (IOException ex) {}
88          JOptionPane.showMessageDialog(null, "Client: Bye!");
89      }
90
91      public static void main (String args[]) throws IOException {
92          ClientBidirectionalSwing client = new ClientBidirectionalSwing();
93          new Thread(client).start();
94      }
95  }
```

Clasa ServerBidirectionalSwing mosteneste si extinde clasa ElementeComuneClientServer:

```
 1  import java.net.*;
 2  import java.io.*;
 3  import javax.swing.JOptionPane;
 4  import java.awt.*;
 5  import java.awt.event.*;
 6  import javax.swing.*;
 7  public class ServerBidirectionalSwing extends ElementeComuneClientServer
 8                                        implements Runnable {
 9     private ServerSocket serverTCP;
10     private JFrame frame;
11     private JTextField inTextGrafic;   // Intrare - linie text grafica (JtextField)
12     private JTextArea outTextGrafic;   // Iesire - zona text grafica (JtextArea)
13     private JScrollBar vertical;
14     private String sirCitit;
15
16     public ServerBidirectionalSwing() throws IOException {
17         super("Server");
18         serverTCP = new ServerSocket(portTCP);             // Creare socket server
19         socketTCP = serverTCP.accept();                    // Creare socket
20         conexiune = new ConexiuneRetea(socketTCP);
21
22         // Crearea obiectului cadru (frame), cu titlu specificat
23         JFrame frame = new JFrame("Server bidirectional Swing"); // Stabilire titlu
24         Container containerCurent = frame.getContentPane();
25         containerCurent.setLayout(new BorderLayout());
26
27         outTextGrafic = new JTextArea(5, 40);  // Zona text non-editabila de iesire
28         JScrollPane scrollPane = new JScrollPane(outTextGrafic,
29                     JScrollPane.VERTICAL_SCROLLBAR_ALWAYS,
30                     JScrollPane.HORIZONTAL_SCROLLBAR_ALWAYS);
31         vertical = scrollPane.getVerticalScrollBar();
32         containerCurent.add("Center", scrollPane);
33         outTextGrafic.setEditable(false);
34         outTextGrafic.append("Pentru oprire introduceti '.' si <Enter>\n\n");
35
36         inTextGrafic = new JTextField(40);      // Camp de text editabil de intrare
37         containerCurent.add("South", inTextGrafic);
38         inTextGrafic.requestFocus(); // Cerere focus pe intrarea text
39
40         // Inregistrare "ascultator" de "evenimente actionare" la "obiectul sursa"
41         inTextGrafic.addActionListener(new AscultatorInText());
42
43         // Iesire din program la inchiderea ferestrei
44         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
45         frame.pack();      // Impachetarea (compactarea) componentelor in container
46         frame.setVisible(true);      // Fereastra devine vizibila
47     }
48
49     // Clasa interna "ascultator" de "evenimente actionare"
50     // implementeaza interfata ActionListener
51     class AscultatorInText implements ActionListener {
52
53       // Tratarea actionarii intrarii de text (introducerii unui "Enter")
54       public void actionPerformed(ActionEvent ev) {
55
56          sirCitit = inTextGrafic.getText();  // Citirea unei linii din intrarea text
57          inTextGrafic.setText("");           // Golirea intrarii text
58          conexiune.printLine(sirCitit);
59
60          outTextGrafic.append("S-a introdus: "+sirCitit+"\n"); // Scriere linie
61
62          // actualizeaza pozitia barei de defilare, varianta imbunatatita a
63          // vertical.setValue(vertical.getMaximum() - vertical.getVisibleAmount());
64          SwingUtilities.invokeLater(new Runnable()
65            { public void run() { vertical.setValue(vertical.getMaximum()); } });
66
67          if (sirCitit.equals(new String(".")))   System.exit(0); // Conditie oprire
68       }
69     }
70
```

```
71      public void run () {
72          String mesaj;
73          while (true) {
74            mesaj = conexiune.nextLine();
75
76            outTextGrafic.append("S-a primit:    " + mesaj + "\n");
77            // actualizeaza pozitia barei de defilare, varianta imbunatatita a
78            // vertical.setValue(vertical.getMaximum() - vertical.getVisibleAmount());
79            SwingUtilities.invokeLater(new Runnable()
80              { public void run() { vertical.setValue(vertical.getMaximum()); } });
81
82            if (mesaj.equals(".")) break;    // Testarea conditiei de oprire
83          }
84          try {  socketTCP.close();        // Inchiderea socketului si a fluxurilor
85          } catch (IOException ex) {}
86          JOptionPane.showMessageDialog(null, "Server: Bye!");
87      }
88
89      public static void main (String args[]) throws IOException {
90          ServerBidirectionalSwing server = new ServerBidirectionalSwing();
91          new Thread(server).start();
92      }
93  }
```



Nu uitati: Daca bara de stare a executiei este activa ( ) verificati cu Alt+Tab daca a aparut o fereastra Java (in spatele ferestrelor vizibile).

Nu uitati: Pentru a opri executia, *right click* pe si Reset Machine (sau Ctrl+Shift+Tab).

In laborator:

1. **Lansati in executie BlueJ. Inchideti proiectele anterioare. Creati un proiect *socketSwing***
2. **In proiectul *socket* creati cele 4 clase ale caror coduri sunt date mai sus. Compilati codurile.**
3. **Testati sistemul ca la laboratorul anterior.**

In laborator:

1. **La unul dintre calculatoare** ***right-click*** pe clasa **ServerBidirectionalSwing**.
2. **Selectati** si **executati main(). Folositi** numarul de port **3000**.

3. **La un alt calculator** (**daca nu aveti la dispozitie un alt calculator in retea, deschideti inca o sesiune BlueJ**) ***right-click*** pe clasa **ClientBidirectionalSwing**, selectati si **executati main()**.
4. **Folositi adresa primului calculator**, pe care se executa **ServerBidirectionalSwing** (**adresa `"localhost"` in cazul in care folositi doua sesiuni BlueJ pe acelasi calculator**), si numarul de port **3000**.
5. **Urmariti efectul** la client si la server.

6.3.2. Sistem de comunicatie client-server (N-la-N) multifilar, bazat pe socket-uri TCP si interfata grafica Swing

Clasa utilitara DialogUtilizator incapsuleaza tratarea interactiunii cu utilizatorul (GUI):

```
 1  import javax.swing.JOptionPane;
 2  import java.awt.*;
 3  import java.awt.event.*;
 4  import javax.swing.*;
 5
 6  public class DialogUtilizator extends JFrame {
 7     private JTextField inTextGrafic; // Intrare - linie de text grafica (JtextField)
 8     private JTextArea outTextGrafic;  // Iesire - zona de text grafica (JtextArea)
 9     private JScrollBar vertical;
10     private ConexiuneRetea conexiune;
11     private String sirCitit;
12
13     // Initializari grafice
14     public DialogUtilizator(String nume) {          // constructor
15         super(nume);        // Stabilire titlu fereastra (JFrame)
16         Container containerCurent = this.getContentPane();
17         containerCurent.setLayout(new BorderLayout());
18         // Zona de text non-editabila de iesire (cu posibilitati de defilare)
19         outTextGrafic = new JTextArea(5, 40);
20         JScrollPane scrollPane = new JScrollPane(outTextGrafic,
21                     JScrollPane.VERTICAL_SCROLLBAR_ALWAYS,
22                     JScrollPane.HORIZONTAL_SCROLLBAR_ALWAYS);
23         vertical = scrollPane.getVerticalScrollBar();
24         containerCurent.add("Center", scrollPane);
25         outTextGrafic.setEditable(false);
26         // Camp de text editabil de intrare
27         inTextGrafic = new JTextField(40);
28         containerCurent.add("South", inTextGrafic);
29         // Inregistrarea "ascultatorului" de "evenimente actionare" la
30         // "obiectul sursa" intrare de text
31         inTextGrafic.addActionListener(new AscultatorInText());
32         // Iesire din program la inchiderea ferestrei
33         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
34         pack();            // Impachetarea (compactarea) componentelor in container
35         setVisible(true);            // Fereastra devine vizibila
36         inTextGrafic.requestFocus(); // Cerere focus pe intrarea de text    }
37     public String nextLine() {
38         String linie = inTextGrafic.getText();     // Citirea unei linii text
39         inTextGrafic.setText("");                  // Golirea intrarii text
40         return linie;
41     }
42     public int nextInt() {
43         String linie = inTextGrafic.getText();     // Citirea unei linii text
44         inTextGrafic.setText("");                  // Golirea intrarii text
45         return Integer.parseInt(linie);
46     }
47     public void printLine(String text) {
48         outTextGrafic.append(text + "\n");
49         vertical.setValue(vertical.getMaximum() - vertical.getVisibleAmount());
50         validate(); repaint();
51     }
52     public void setConexiune(ConexiuneRetea conexiune) {
53         this.conexiune = conexiune;
54     }
55
56     // Clasa interna "ascultator" de "evenimente actionare"
57     // implementeaza interfata ActionListener
58     class AscultatorInText implements ActionListener {
59         // Tratarea actionarii intrarii de text (introducerii unui "Enter")
60         public void actionPerformed(ActionEvent ev) {
61            sirCitit = nextLine();
62            conexiune.printLine(sirCitit);
63            if (sirCitit.equals(new String("BYE"))) System.exit(0); //Conditie oprire
64         }
65     }
66  }
```

Clasa utilitara ConexiuneRetea incapsuleaza tratarea conexiunilor TCP:

```
 1  import java.net.*;
 2  import java.io.*;
 3  import java.util.Scanner;
 4
 5  public class ConexiuneRetea {
 6     private Socket conexiune;
 7     private Scanner scannerTCP;
 8     private PrintStream printerTCP;
 9     public ConexiuneRetea(Socket conexiune) throws IOException {
10         this.conexiune = conexiune;
11         this.scannerTCP = new Scanner(conexiune.getInputStream());
12         this.printerTCP = new PrintStream(conexiune.getOutputStream());
13     }
14     public String nextLine() {
15         return this.scannerTCP.nextLine();
16     }
17     public int nextInt() {
18         return this.scannerTCP.nextInt();
19     }
20     public void printLine(String text) {
21         this.printerTCP.println(text);
22         this.printerTCP.flush();
23     }
24     public String getLocalAddress() {
25         return socket.getLocalAddress().getHostAddress();
26     }
27     public String getRemoteAddress() {
28         return socket.getInetAddress().getHostAddress();
29     }
30  }
31 }
```

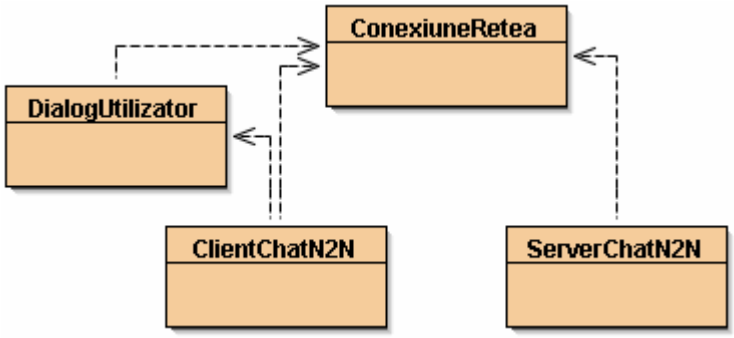
Clasa ServerChatN2N reprezinta serverul TCP care poate trata mai multi clienti:

```
 1  import java.net.*;
 2  import java.io.*;
 3  import javax.swing.JOptionPane;
 4
 5  public class ServerChatN2N extends Thread {
 6     private static int numarClienti = 0;
 7     private static DialogUtilizator04 dialog;
 8     private static int portTCP;
 9     private static ServerSocket serverTCP;
10     private static Socket socketTCP;
11     public static final int NUMAR_MAXIM_CLIENTI = 10;
12     private static ConexiuneRetea03[] conexiuni;
13     private int numarulConexiuniiCurente; // incepe cu 0
14     public ServerChatN2N(int numarulConexiuniiCurente) throws IOException {
15         this.numarulConexiuniiCurente = numarulConexiuniiCurente;
16     }
17
18     public static void main (String args[]) throws IOException {
19         dialog = new DialogUtilizator04("SERVER CHAT N2N");
20         conexiuni = new ConexiuneRetea03[NUMAR_MAXIM_CLIENTI];
21         portTCP = Integer.parseInt(JOptionPane.showInputDialog(
22                     "Introduceti numarul de port al serverului"));
23         serverTCP = new ServerSocket(portTCP); // Creare socket server
24         while (true) {
25             socketTCP = serverTCP.accept();  // Creare socket
26             conexiuni[numarClienti] = new
27                                 ConexiuneRetea03(socketTCP);
28             ServerChatN2N server = new ServerChatN2N(numarClienti++);
29             server.start();
30         }
31     }
32     public void removeConnexion(int crt) {
33         for (int i=crt; i<numarClienti-1; i++) {
34             conexiuni[i] = conexiuni[i+1];
35         }
36         numarClienti--;
37     }
```

```
38      public void run() {  // Fir executie receptie prin socket
39          String mesaj;
40          while(true) {
41              mesaj = this.conexiuni[numarulConexiuniiCurente].nextLine();
42              // this.dialog.printLine(mesaj);
43              for (int i=0; i<numarClienti; i++) {
44                  conexiuni[i].printLine(mesaj);
45              }
46              // Testarea conditiei de oprire
47              if (mesaj.equals("BYE")) {
48                  removeConnexion(numarulConexiuniiCurente);
49                  this.dialog.printLine("Bye!");
50                  this.stop();
51              }
52          }
53      }
54  }
```

Clasa ClientChatN2N reprezinta clientul TCP cu interfata grafica:

```
 1   import java.net.*;
 2   import java.io.*;
 3   import javax.swing.JOptionPane;
 4
 5   public class ClientChatN2N extends Thread {
 6      private DialogUtilizator04 dialog;
 7      private ConexiuneRetea03 conexiune;
 8      private Socket socketTCP;
 9      private int portTCP;
10      private InetAddress adresaIP;
11
12      public ClientChatN2N() throws IOException {
13          dialog = new DialogUtilizator04("CLIENT CHAT N2N");
14          portTCP = Integer.parseInt(JOptionPane.showInputDialog(
15                      "Introduceti numarul de port al serverului"));
16          adresaIP = InetAddress.getByName(JOptionPane.showInputDialog(
17                      "Introduceti adresa IP a serverului"));
18          socketTCP = new Socket(adresaIP, portTCP); // Creare socket
19          conexiune = new ConexiuneRetea03(socketTCP);
20          dialog.printLine("\nPentru oprire introduceti \"BYE\" si <Enter>\n");
21      }
22
23      public static void main (String args[]) throws IOException {
24          ClientChatN2N client = new ClientChatN2N();
25          client.start();
26          String mesaj;
27          while(true) {
28              mesaj = client.dialog.nextLine("Introduceti mesajul");
29              client.conexiune.printLine(mesaj);
30              if (mesaj.equals("BYE")) break;  // Testarea conditiei de oprire
31          }
32          try {
33              client.socketTCP.close(); // Inchidere socket (implicit fluxuri)
34          } catch (IOException ioe) {};
35          client.dialog.printLine("Bye!");
36          System.exit(0);
37      }
38
39      public void run() {  // Fir executie receptie prin socket
40          String mesaj;
41          while(true) {
42              mesaj = this.conexiune.nextLine();
43              this.dialog.printLine(conexiune.getLocalAddress() + mesaj);
44              if (mesaj.equals("BYE")) break;     // Testarea conditiei de oprire
45          }
46          try {
47              this.socketTCP.close(); // Inchidere socket (implicit fluxuri)
48          } catch (IOException ioe) {};
49          this.dialog.printLine("Bye!");
50          System.exit(0);
51      }
52  }
```



In laborator:

1. **Lansati in executie BlueJ**. **Inchideti proiectele anterioare** (**Ctrl+W**). **Creati un proiect *chatn2n***
2. **Creati cele 4 clase ale caror coduri sunt date mai sus**. **Compilati codurile.**

Optional, in laborator:

1. La unul dintre calculatoare *right-click* pe clasa ServerChatN2N. Selectati si executati main(). Folositi numarul de port 4000. La alte 2-3 calculatoare (daca nu aveti la dispozitie un alt calculator in retea, deschideti mai multe sesiuni BlueJ) *right-click* pe clasa Client ChatN2N, selectati si executati main(). Folositi adresa primului calculator (pe care se executa ServerChatN2N) si numarul de port 4000.

6.4. Precizari privind mini-proiectul pentru examen

Tema mini-proiectului este: Sistem de comunicatie (avansat, pentru bonus) client-server (N-la-N) bazat pe socket TCP, fire de executie si interfata grafica Swing pornind de la codurile sursa date ca exemplu la acest laborator (NU de la zero!).

Proiectul va contine (pentru fiecare grup de studenti):

- **dosarul cu documentatia proiectului** (**6-8 pagini**), incluzand:
  - **titlul proiectului si autorii** – cu indicarea rolului fiecaruia - **prima pagina**
  - **descrierea claselor si metodelor utilizate (rol, structura, comportament) - 3 pagini**
  - **documentarea solutiilor tehnice care nu apar in exemple, a problemelor intampinate,**
  - **descrierea modului de utilizare al sistemului** (cu *screenshot*-uri) **– 2 pagini**
- **listingurile codurilor sursa** atasate ca **anexa la dosar**
- **proiectul in forma electronica** (**pe CD, atasat de dosar cu un plic**), continand:
  - **documentatia,**
  - **codurile sursa ale proiectului** si
  - **codurile sursa ale temelor de casa date la lucrarile de laborator 1, 2, 3 si 4**.

Sustinerea proiectului presupune:

- **predarea dosarului cu documentatia proiectului (avand atasata forma electronica)**
- **prezentarea executiei sistemului daca acesta include facilitati noi, avansate (pentru bonus)**
- **prezentarea rolului fiecarui autor la realizarea sistemului si a documentarii lui**
- **prezentarea solutiilor tehnice noi (pentru bonus)**
- **raspunsuri la intrebari privind detalii ale proiectului (din documentatie si listinguri)**

Imbunatatiri / modificari posibile (pentru bonus) de la simplu la complex:

- adaugarea unui **buton BYE**
- **interfata grafica si la server** (ca la client)
- **imbunatatiri ale tratarii exceptiilor**
- **canal privat** intre 2 clienti prin analiza lexicala
- **canal privat** intre 2 clienti prin socket-uri noi
- **lista utilizatori prosibili**, la server, si **autentificare cu nume si parola**
- **interfata grafica de monitorizare la server**
- **inlocuirea tabloului** conexiunilor **cu Vector**
- **logfile** la clienti si / server pt. memorare dialog
- **atasarea unor nickname-uri** utilizatorilor
- **difuzarea nickname-uri** utilizatorilor si **afisarea unei liste participantilor**
- **invitarea / anuntarea** potentialilor utilizatori
- inlocuirea socketurilor TCP cu **socketuri UDP**
- trecerea de la client-server la un **sistem descentralizat**