2009 - **2010**

Tehnologii de Programare in Internet (TPI / RST)

Titulari curs: **Mihnea Magheti**, Eduard-Cristian Popovici

Suport curs: *http://discipline.elcom.pub.ro/tpi/*

Moodle: *http://electronica07.curs.ncit.pub.ro/course/category.php?id=3*

Structura cursului

Continut curs TPI

1. Introducere in tehnologiile Internet

2. Introducere in tehnologiile desktop (SE) Java

- 2.1. Elemente de baza. Tipuri de date referinta. Clase de biblioteca
- 2.2. Clase pentru fluxuri de intrare-iesire (IO)

3. Programarea la nivel socket in Java

- 3.1. Introducere in Protocolul Internet (IP) si stiva de protocoale IP
- 3.2. Socketuri flux (TCP) Java si programe multifilare (threads)
- 3.3. Socketuri datagrama (UDP) Java

4. Tehnologii Java de programare a aplicatiilor Web (EE) Java

- 4.1. Tehnologii client. Miniaplicatii Java (applet-uri)
- 4.2. Clase pentru interfete grafice cu utilizatorul (AWT, Swing)
- 4.3. Platforma Java EE. Arhitectura si tehnologiile implicate
- 4.4. Tehnologii server. Tehnologia Java Servlet
- 4.5. Tehnologia Java ServerPages (JSP)
- 4.6. Accesul la baze de date prin tehnologii Java (JDBC, Hibernate)
- 4.7. Tehnologii avansate (frameworks, componente EJB, Servicii Web)

3. Programarea la nivel socket in Java

3.3. Socketuri datagrama (UDP) Java

3.3. Socketuri datagrama (UDP) Java

Crearea si utilizarea socket-urilor UDP

Clasa DatagramPacket reprezinta

- un **pachet UDP** (o **datagrama**)

  - utilizat pentru **livrare fara conexiune** (include in mod normal informatii privind adresele IP si porturile sursa si destinatie)

Clasa DatagramSocket reprezinta

- un ***socket* UDP**

  - **prin care se trimit sau se primesc pachete datagrama** peste retele IP

    - **prin intermediul protocolului UDP**

Clasa MulticastSocket (subclasa a DatagramSocket care adauga functionalitati legate de multicast) poate fi utilizata pentru

- **trimiterea si receptia** unui **pachet datagrama** catre, respectiv dinspre, un **grup multicast**

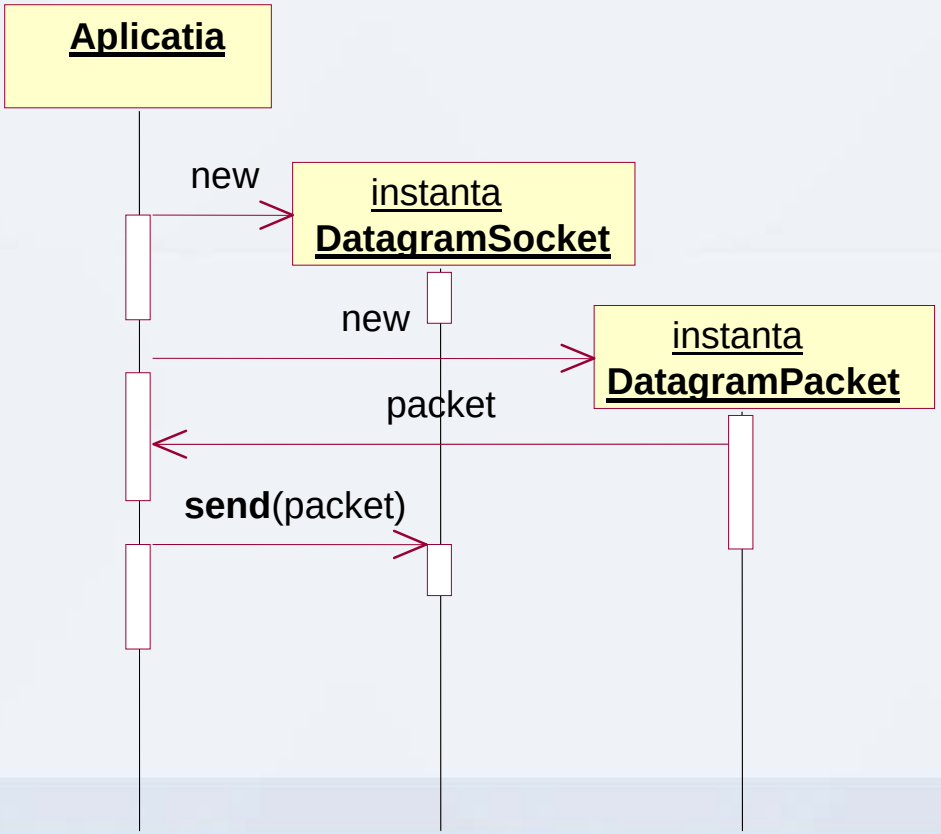
3.3. Socketuri datagrama (UDP) Java

Crearea si utilizarea socket-urilor UDP

Un DatagramPacket este **trimis printr-un DatagramSocket**

- **apeland** la metoda **send()** a clasei **DatagramSocket**

- careia i **se paseaza ca parametru** respectivul **DatagramPacket**



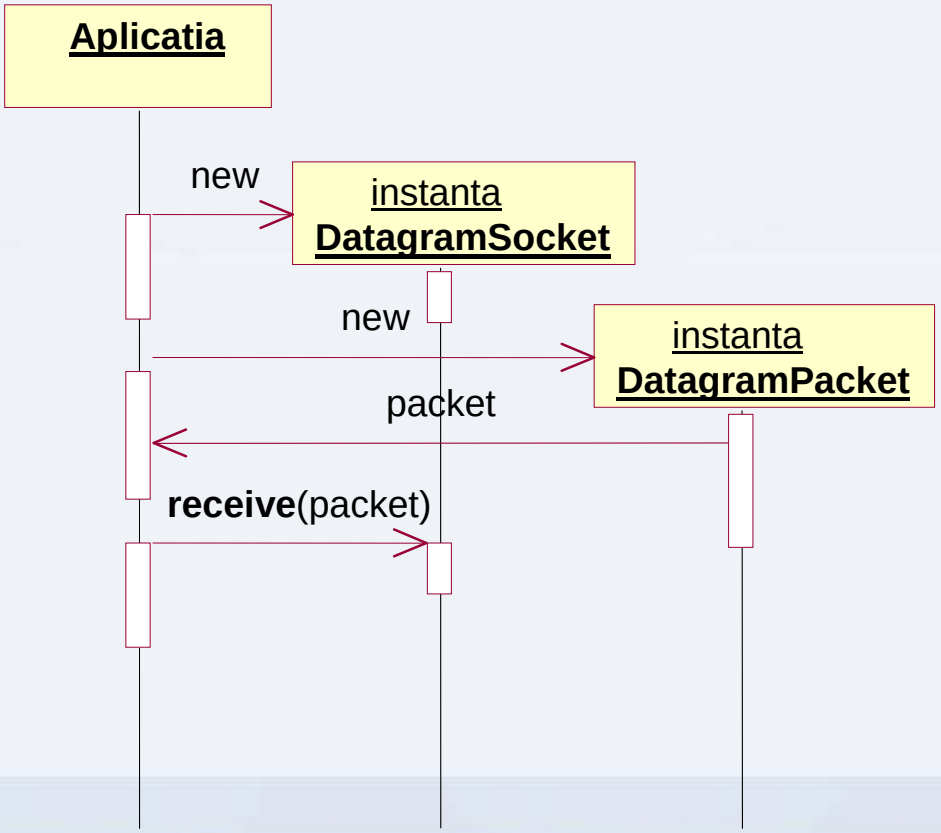
3.3. Socketuri datagrama (UDP) Java

Crearea si utilizarea socket-urilor UDP

Un DatagramPacket este **primit printr-un DatagramSocket**

- **apeland** la metoda **receive()** a clasei **DatagramSocket**

  - careia i **se paseaza ca parametru** un **DatagramPacket** pregatit pentru receptie



3.3. Socketuri datagrama (UDP) Java

Crearea si utilizarea socket-urilor UDP

Principalii constructori ai clasei DatagramPacket

DatagramPacket(byte[] buf, int length) Construieste un obiect DatagramPacket pregatindu-l pentru **receptia unui pachet** de lungime length, furnizandu-i tabloul de octeti buf in care sa fie plasate datele pachetului (length trebuie sa fie mai mic sau egal cu buf.length).
DatagramPacket(byte[] buf, int length, InetAddress address, int port) Construieste un obiect DatagramPacket pregatindu-l pentru **trimiterea unui pachet** de lungime length catre numarul de port UDP specificat (port) al gazdei cu adresa specificata (address), furnizandu-i tabloul de octeti buf din care sa fie preluate datele pachetului (length trebuie sa fie mai mic sau egal cu buf.length).

Principalii constructori ai clasei DatagramSocket

DatagramSocket() Construieste un *socket* datagrame si il leaga la un port UDP disponibil pe masina locala.
DatagramSocket(int port) Construieste un *socket* datagrame si il leaga la portul UDP specificat pe masina locala.
DatagramSocket(int port, InetAddress laddr) Construieste un *socket* datagrame legat pe local la portul UDP si adresa IP specificate

3.3. Socketuri datagrama (UDP) Java

Crearea si utilizarea socket-urilor UDP

Declaratiile si descrierea catorva metode ale clasei **`DatagramPacket`**

InetAddress	getAddress() Returneaza **adresa IP** sub forma de obiect InetAddress a masinii catre care pachetul curent va fi trimis sau de la care va fi receptionat.
byte[]	getData() Returneaza tabloul de octeti (*bufferul*) care contine **datele** pachetului
int	getLength() Returneaza numarul de octeti (**lungimea** *bufferului*) de date de trimis sau de receptionat.
int	getOffset() Returneaza indexul (***offsetul***) la care sunt plasate datele de trimis sau de receptionat in tabloul de octeti.
int	getPort() Returneaza **numarul de port UDP** al masinii catre care pachetul curent va fi trimis sau de la care pachetul curent va fi receptionat.

3.3. Socketuri datagrama (UDP) Java

Crearea si utilizarea socket-urilor UDP

Declaratiile si descrierea catorva metode ale clasei **`DatagramPacket`**

void	**setAddress**(InetAddress iaddr) Stabileste **adresa IP** sub forma de obiect InetAddress a masinii catre care pachetul curent va fi trimis.
void	**setData**(byte[] buf) Stabileste tabloul de octeti (*bufferul*) care contine **datele** pachetului curent (implicit indexul offset=0).
void	**setData**(byte[] buf, int offset, int length) Stabileste tabloul de octeti (*bufferul*) care contine **datele** pachetului curent specificandu-i indexul offset de la care sa inceapa plasarea datelor pachetului in tablou.
void	**setLength**(int length) Stabileste numarul de octeti (**lungimea** *bufferului*) de date de trimis sau de receptionat.
void	**setPort**(int iport) Stabileste **numarul de port UDP** al masinii catre care pachetul va fi trimis.

3.3. Socketuri datagrama (UDP) Java

Crearea si utilizarea socket-urilor UDP

Declaratiile si descrierea catorva metode ale clasei **`DatagramSocket`**

InetAddress	getLocalAddress() Obtine adresa IP locala la care este legat *socketul* curent.
int	getLocalPort() Returneaza numarul de port local la care este legat *socketul* curent.
void	receive (DatagramPacket p) **Receptioneaza un pachet datagrama** prin *socketul* curent.
void	send (DatagramPacket p) **Trimite un pachet datagrama** prin *socketul* curent.
void	close() Inchide *socketul* datagrama curent.
void	connect (InetAddress address, int port) **"Conecteaza" *socketul* curent la adresa IP si numarul de port specificate (acestea actioneaza ca un filtru, prin *socketul* conectat putand fi trimise sau primite pachete doar catre respectiv de la adresa IP si numarul de port specificate).** Implicit *socketul* e neconectat.

3.3. Socketuri datagrama (UDP) Java

Crearea si utilizarea socket-urilor UDP

Declaratiile si descrierea catorva metode ale clasei **`DatagramSocket`**

InetAddress	**getInetAddress**() Returneaza adresa IP la care *socketul* curent e conectat (null daca nu e).
int	**getPort**() Returneaza numarul de port UDP la care *socketul* curent este conectat (-1 daca nu este conectat).
void	**disconnect**() Deconecteaza *socketul* curent.
boolean	**isConnected**() Returneaza o valoare logica indicand daca *socketul* curent e conectat
int	**getSoTimeout**() Obtine valoarea optiunii SO_TIMEOUT.
void	**setSoTimeout**(int timeout) Stabileste valoarea optiunii SO_TIMEOUT la un *timeout* specificat, in millisecunde.

3.3. Socketuri datagrama (UDP) Java

Crearea si utilizarea socket-urilor UDP

Declaratiile si descrierea catorva metode ale clasei **`DatagramSocket`**

int	**getReceiveBufferSize**() Returneaza valoarea optiunii SO_RCVBUF pentru *socketul* curent, adica dimensiunea *bufferului* utilizat de platforma pentru pachetele primite de *socket*
void	**setReceiveBufferSize**(int size) Stabileste optiunea SO_RCVBUF pentru *socket* la valoarea specificata.
int	**getSendBufferSize**() Returneaza valoarea optiunii SO_ SNDBUF pentru *socketul* curent, adica dimensiunea *bufferului* utilizat de platforma pentru pachetele trimise de *socket*
void	**setSendBufferSize**(int size) Stabileste optiunea SO_SNDBUF pentru *socket* la valoarea specificata.
int	**getTrafficClass**() Obtine valoarea octetului clasa de trafic (QoS) sau tip de serviciu (ToS) in antetul datagramei IP care incapsuleaza datagramele UDP trimise prin *socket*
void	**setTrafficClass**(int tc) Stabileste valoarea octetului clasa de trafic (QoS) sau tip serviciu (ToS) in antetul datagramei IP care incapsuleaza datagramele UDP trimise prin *socket*

Crearea si utilizarea socket-urilor UDP

Secventa tipica pentru **crearea unui pachet datagrama (UDP) pentru timitere**

```
1       // Stabilirea adresei serverului
2       String adresaIP = "localhost";
3
4       // Stabilirea portului serverului
5       int portUDP = 2000;
6
7       // Crearea tabloului de octeti (bufferului) de date
8       String mesaj = "mesaj de trimis";
9       byte[] bufferDate = mesaj.getBytes();
10
11      // Crearea pachetului de trimis
12      try {
13        DatagramPacket pachetUDP = new DatagramPacket(bufferDate,
14          bufferDate.length, InetAddress.getByName(adresaIP), portUDP);
15      }
16      catch (UnknownHostException ex) {
17        System.err.println(ex);
18      }
```

3.3. Socketuri datagrama (UDP) Java

Crearea si utilizarea socket-urilor UDP

Secventa tipica pentru **crearea unui pachet datagrama (UDP) pentru receptie**

```
1       // Crearea tabloului de octeti (bufferului) de date
2       byte[] bufferDate = new byte[4096];
3
4       // Crearea pachetului de primit
5       try {
6         DatagramPacket pachetUDP = new DatagramPacket(bufferDate,
7                                                       bufferDate.length);
8       }
9       catch (UnknownHostException ex) {
10        System.err.println(ex);
11      }
```

Secventa tipica pentru **citirea din pachetul UDP a datelor primite**

```
1       // Obtinerea datelor pachetului ca tablou de octeti
2       bufferDate = pachetUDP.getData();
3
4       // Reconstructia mesajului text
5       String mesajPrimit =
6               new String(bufferDate, 0, pachetUDP.getLength());
```

3.3. Socketuri datagrama (UDP) Java

Crearea si utilizarea socket-urilor UDP

Secventa tipica pentru **crearea socket-ului unei aplicatii client**

```
// Crearea socketului
DatagramSocket socketUDP = new DatagramSocket();
```

Secventa tipica pentru **crearea socket-ului unei aplicatii server**

```
// Stabilirea portului serverului
int portServer = 2000;
// Crearea socketului
DatagramSocket socketUDP = new DatagramSocket(portServer);
```

Trimiterea de date prin socket

```
// Trimiterea pachetului UDP
socketUDP.send(pachetDeTrimis);
```

Primirea de date prin socket

```
// Receptia unui pachet UDP - blocare in asteptare
socketUDP.receive(pachetPrimit);
```

3.3. Socketuri datagrama (UDP) Java

Crearea si utilizarea socket-urilor UDP

Identificarea porturilor UDP locale pe care sunt conexiuni prin incercarea de a crea pe ele socket-uri UDP

```
 1 import java.net.*;
 2
 3 public class ScannerPorturiUDPLocale {
 4   public static void main (String args[]) {
 5     DatagramSocket serverUDP;
 6     for (int portUDP=1; portUDP < 65536; portUDP++) {
 7       try {
 8         // Urmatoarele linii genereaza exceptie prinsa de blocul catch
 9         // in cazul in care e deja un server pe portul portUDP
10         serverUDP = new DatagramSocket(portUDP);
11         serverUDP.close();
12       }
13       catch (SocketException ex) {
14         System.err.println("Exista server UDP pe portul " + portUDP);
15       }
16     }
17   }
18 }
```

3.3. Socketuri datagrama (UDP) Java

Crearea si utilizarea socket-urilor UDP

Program server ecou cu pierderi (10% din pachetele primite)

```
 1  import java.net.*;
 2  import java.io.*;
 3
 4  /**
 5   *  Server ecou care pierde in mod aleator 10% din pachete
 6   */
 7  public class ServerEcouCuPierderiDatagrame {
 8
 9    public static void main(String args[]) throws IOException {
10
11      BufferedReader inConsola  = new BufferedReader(new
12                                      InputStreamReader(System.in));
13
14      System.out.print("Introduceti numarul de port al serverului: ");
15      int portUDP = Integer.parseInt(inConsola.readLine());
16
17      // Crearea socket-ului
18      DatagramSocket socketUDP = new DatagramSocket(portUDP);
```

Crearea si utilizarea socket-urilor UDP

Program server ecou cu pierderi (10% din pachetele primite)

```
19      // Tratarea clientului
20      while (true) {
21
22        // Primirea pachetului
23        byte bufferDate[] = new byte[65535];
24
25        DatagramPacket pachetUDP = new DatagramPacket(bufferDate,
26                                                      bufferDate.length);
27
28        socketUDP.receive(pachetUDP);
29
30        System.out.println("S-au primit " + pachetUDP.getLength() +
31                           " octeti, <<" + pachetUDP.getData() + ">>");
32
33        // Decizia asupra modului de tratare a pachetului
34        // Trimiterea pachetului
```

Crearea si utilizarea socket-urilor UDP

Program server ecou cu pierderi (10% din pachetele primite)

```
35          if (Math.random() < .9) {
36            System.out.println("Pachetul va fi trimis in ecou.");
37
38            DatagramPacket pachetUDPEcou = new DatagramPacket(
39                        pachetUDP.getData(), pachetUDP.getLength(),
40                        pachetUDP.getAddress(), pachetUDP.getPort());
41
42            socketUDP.send(pachetUDPEcou);
43            System.out.println("Raspuns trimis.");
44          }
45
46          // Aruncarea pachetului
47          else {
48            System.out.println("Pachetul a fost aruncat.");
49          }
50        }
51      }
52  }
```

Crearea si utilizarea socket-urilor UDP

Client pentru server de ecou cu pierderi (retransmite la intarzieri >10 secunde ale pachetelor ecou de la server)

```
 1  import java.net.*;
 2  import java.io.*;
 3
 4  public class ClientRetransmisie {
 5    protected DatagramSocket socketUDP;
 6    protected DatagramPacket pachetUDP;
 7    protected Asteptare timer;
 8
 9    protected void trimitePachet() throws IOException {
10
11      socketUDP.send(pachetUDP);
12      System.out.println("Pachet trimis.");
13
14      // Crearea firului timer Asteptare
15      timer = new Asteptare(10, this);
16
17      // Pornirea firului timer Asteptare
18      timer.start();
19    }
```

Crearea si utilizarea socket-urilor UDP

Client pentru server de ecou cu pierderi

```
20    protected void primestePachet() throws IOException {
21
22      byte bufferDate[] = new byte[65535];
23
24      DatagramPacket pachetUDP = new DatagramPacket(bufferDate,
25                                                    bufferDate.length);
26      socketUDP.receive(pachetUDP);
27
28      ByteArrayInputStream inTablou = new ByteArrayInputStream(
29                          pachetUDP.getData(), 0, pachetUDP.getLength());
30
31      DataInputStream inDateFormatate = new DataInputStream(inTablou);
32
33      String textPrimit = inDateFormatate.readUTF();
34      System.out.println("S-a primit " + textPrimit);
35
36      // Oprirea firului timer
37      timer.stop();
38    }
```

Crearea si utilizarea socket-urilor UDP

Client pentru server de ecou cu pierderi

```
39    public static void main(String args[]) throws IOException {
40      String adresaServer = JOptionPane.showInputDialog(
41                            "Introduceti adresa IP a serverului: ");
42      int portServer = Integer.parseInt(JOptionPane.showInputDialog(
43                       "Introduceti numarul de port al serverului: "));
44      String textDeTrimis = JOptionPane.showInputDialog(
45      ClientRetransmisie client = new ClientRetransmisie();
46
47      // Creare socket
48      client.socketUDP = new DatagramSocket();
49
50      // Constructie pachet
51      ByteArrayOutputStream outTablou = new ByteArrayOutputStream();
52      DataOutputStream outDateFormatate = new DataOutputStream(outTablou);
53      outDateFormatate.writeUTF(textDeTrimis);
54
55      byte[] bufferDate = outTablou.toByteArray();
56      client.pachetUDP = new DatagramPacket(bufferDate, bufferDate.length,
57                         InetAddress.getByName(adresaServer), portServer);
```

Crearea si utilizarea socket-urilor UDP

Client pentru server de ecou cu pierderi

```
58      while (true) {
59
60        // Trimitere pachet
61        client.trimitePachet();
62
63        // Primire pachet
64        client.primestePachet();
65
66        System.out.println("Pauza 2 secunde...");
67        try {
68          Thread.sleep(2000);
69
70        } catch (InterruptedException ex) {
71          ex.printStackTrace();
72        }
73      }
74    }
75  }
```

3.3. Socketuri datagrama (UDP) Java

Crearea si utilizarea socket-urilor UDP

Clientul utilizeaza urmatoarea clasa care lucreaza cu fire de executie

```
 1    import java.net.*;
 2    import java.io.*;
 3
 4    class Asteptare extends Thread {
 5      private ClientRetransmisie client;
 6      private int durata;
 7
 8      public Asteptare (int durata, ClientRetransmisie client) {
 9        this.client = client;
10        this.durata = durata;
11      }
12      public void run() {
13        try {
14          Thread.sleep (durata*1000);
15          System.out.println("Retransmisie...");
16          try {
17            // Retransmisie pachet
18            client.trimitePachet();
19          } catch (IOException ex) { ex.printStackTrace();  }
20        } catch (InterruptedException ex) { ex.printStackTrace ();  }
21      }
22   }
```