9 Kommunikation i Java

Skansholm kap 18

- adresser
- filaccess över nätverk
- portar och sockets
- datagram
- multicasting
- client-server

Adressering

- En **URL** (=Uniform Resource Locator) identifierar en resurs på **WWW**

  protokoll`://`**adress**`:`**port**`/`**filnamn**

  `http://www.chalmers.se/dit/index.htm`
- En **IP-adress** identifierar en dator på **Internet**
  - `129.16.214.8`
  - I textform som ovan eller som en följd av bytes

Java-klasser för Adressering

- `java.net.URL`
  - för hantering av URL-objekt
- `java.net.URLConnection`
  - för hantering av resurser som refereras av URL-objekt
- `java.net.InetAddress`
  - för hantering av IP-adresser

`java.net.URL`

Metoder (i urval)	Förklaring
`new URL(String adress)`	skapar ett URL-objekt av `adress`
`URLConnection openConnection()`	öppnar en förbindelse
`String getProtocol()`	ger protokollet
`String getHost()`	ger datoradressen
`String getPort()`	ger portnumret
`String getFile()`	ger filnamnet

`java.net.URLConnection`

Metoder (i urval)	Förklaring
`InputStream getInputStream()`	ger en inström som läser från den öppna förbindelsen
`OutputStream getOutputStream()`	ger en utström som skriver på den öppna förbindelsen
`Object getContent()`	ger objekt från strömmen
`int getContentLength()`	ger resursens (filens) längd
`String getContentType()`	ger resursens typ
`long getDate()`	ger tidpunkt för resursens skapelse
`long getLastModified()`	ger tidpunkt för senaste ändring

Ex. Läsning av fil

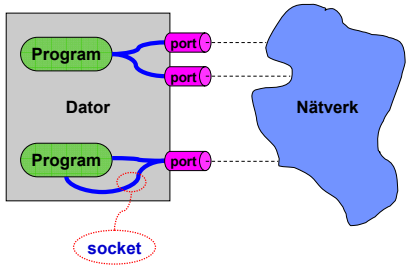
```
try {
    URL url = new URL("http://www.chalmers.se/dir/file.txt");
    URLConnection uc = url.openConnection();
}
catch (MalformedURLException e) { ... }
catch (IOException e) { ... }
try {
   BufferedReader in =
        new BufferedReader(
              new InputStreamReader(uc.getInputStream()));
   String line;
   while ( (line = in.readLine()) != null )
      System.out.println(line);
}
catch (IOException e) { ... }
```

Portar och sockets

- En **port** är en logisk anslutningspunkt till en tjänst som tillhandahålls av en dator.
- En **socket** är en kommunikationskanal mellan ett program och omgivningen genom en port.

Portnumrering
- Portnummer: 0-65535
- 0-1024 är standardiserade, resten kan användas fritt
- 80 http-kommunikation
- 21 ftp-kommunikation

Portar och sockets

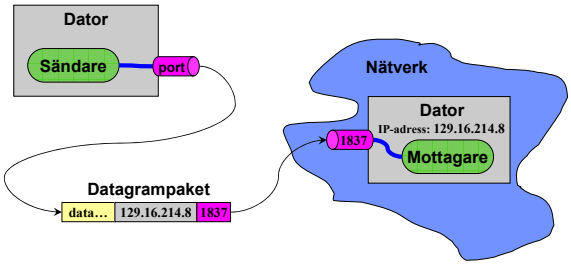


Datagram

- Ett **datagram** är ett datapaket som innehåller
  - data som skall överföras (”nyttolast”)
  - mottagardatorns IP-adress
  - portnumret för mottagartjänsten
- **Datagram** kan komma fram i godtycklig ordning
  - Ingen leveransgaranti

data	129.16.214.8	1837

Datagram



`java.net.InetAddress`

Metoder (i urval)	Förklaring
`InetAddress getLocalHost()`	ger adressen till den egna datorn
`InetAddress getByName(String hostName)`	ger adressen till datorn `hostName` (namnet symboliskt eller IP i textform)
`String getHostName()`	ger namnet på den egna datorn
`String getHostAddress()`	ger IP-adressen i textform till den egna datorn

Symbolikt namn `www.chalmers.se`
IP i textform `129.16.214.8`

`java.net.DatagramSocket`

Metoder (i urval)	Förklaring
`new DatagramSocket(int port)`	skapar en socket genom porten `port`
`new DatagramSocket()`	skapar en socket genom en ledig port
`send(DatagramPacket p)`	sänder datagrammet `p` från denna socket
`receive(DatagramPacket p)`	tar emot datagrammet `p` från denna socket
`setSoTimeOut(long t)`	anger att `receive` skall vänta högst `t` ms vid timeout kastas `SocketTimeoutException`
`close()`	stänger förbindelsen
`int getLocalPort()`	ger numret på den lokala porten som denna socket är bunden till

java.net.DatagramPacket

Metoder (i urval)	Förklaring
`new DatagramPacket(byte[] buf, int len, InetAddress iAddr, int port)`	skapar ett datagrampaket för sändning av `len` bytes i `buf` till `port` på `iAddr`
`new DatagramPacket(byte[] buf, int len)`	skapar ett datagrampaket för mottagning av max `len` bytes i `buf` (`len <= buf.length`)
`byte[] getData()`	ger en referens till paketets databuffert
`int getLength()`	ger längden på paketets datainnehåll
`InetAddress getAddress()`	ger adressen till avsändaren (vid mottagning) ger adressen till mottagaren (vid sändning)
`int getPort()`	ger portnumret till avsändaren (vid mottagning) ger portnumret till mottagaren (vid sändning)

Exempel: Skicka och ta emot datagram

- Programmet **`SendMessage`** läser text från tangentbordet och skickar varje rad som datagram till IP-adressen `127.0.0.1` på port `1234`
- Programmet **`ReceiveMessage`** hämtar datagram från port `1234` och skriver ut dem i konsolfönstret
- 127.0.0.1 är en s.k. loop-back-adress som lämpar sig för testning lokalt innanför brandvägg.

SendMessage.java

```
import java.net.*;
import java.io.*;
class SendMessage {
    public static void main( String[] arg ) throws
        UnknownHostException, SocketException, IOException
    {
        BufferedReader in = new BufferedReader(new InputStreamReader(System.in));

        InetAddress toAddr = InetAddress.getByName(arg[ 0 ]);
        int toPort = Integer.parseInt(arg[ 1 ]);
        DatagramSocket socket = new DatagramSocket();

        while (true) {
            System.out.print("? ");
            String message = in.readLine();
            if (message == null)
                break;
            byte[] data = message.getBytes();
            socket.send(new DatagramPacket(data, data.length, toAddr, toPort));
        }
    }
}
```

exekvering
```
> java SendMessage 127.0.0.1 1234
```


ReceiveMessage.java

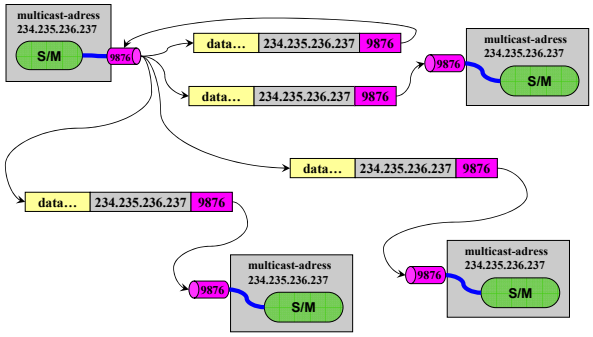
```
import java.net.*;
import java.io.*;
class ReceiveMessage {
    public static void main(String[] arg) throws
            SocketException, IOException
    {
        int receivePort = Integer.parseInt(arg[ 0 ]);
        DatagramSocket socket = new DatagramSocket(receivePort);
        byte[] data = new byte[1024];

        while (true) {
            DatagramPacket packet = new DatagramPacket(data, data.length);
            socket.receive(packet);
            System.out.println("Message from " + packet.getAddress().getHostName());
            String message = new String(packet.getData(), 0, packet.getLength());
            System.out.println(message);
        }
    }
}
```

exekvering
```
> java ReceiveMessage     1234
```


Multicasting

- **multicasting**: ett datagram skickas till alla medlemmar i en grupp (1:N)
- sändaren kan själv ingå i gruppen som får datagrammet
- fiktiva (klass D) IP-adresser mellan `224.0.0.0` och `239.255.255.255` används för adressering
- för multicasting används **`MulticastSocket`**

Multicasting



java.net.MulticastSocket

Metoder (i urval)	Förklaring
`new MulticastSocket(int port)`	skapar en multicast-socket genom porten `port`
`send(DatagramPacket p)`	sänder datagrammet `p` från denna socket
`receive(DatagramPacket p)`	tar emot datagrammet `p` från denna socket
`joinGroup(InetAddress iAddr)`	Ansluter denna socket till multicast-gruppen på adressen `iAddr`
`leaveGroup(InetAddress iAddr)`	Kopplar bort denna socket från multicast-gruppen på adressen `iAddr`
`close()`	stänger förbindelsen

Exempel: Skicka ett Multicast-datagram

```
MulticastSocket sock = new MulticastSocket(9876);
iAddr = InetAddress.getByName(234.235.236.237);
sock.joinGroup(iAddr);
```

Anslut till gruppen

Skicka datagrammet

```
DatagramPacket p =
   new DatagramPacket(data,data.length,iAddr,9876);
sock.send(p);
```

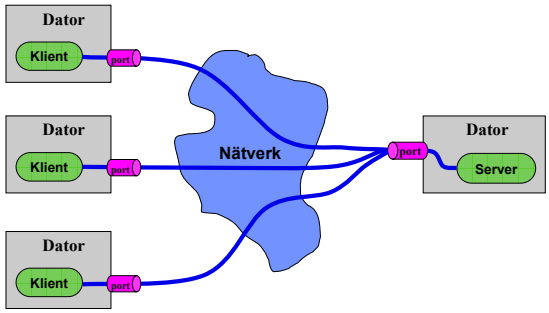
Lämna gruppen

```
sock.leaveGroup(iAddr);
```


Client-server

- Uppkopplad förbindelse mellan två program
- Dataflöde mellan två parter
  - Adressering ej nödvändig vid varje överföring

Client-Server



java.net.Socket

Metoder (i urval)	Förklaring
`new Socket(InetAddress iAddr, int port)`	ansluter till porten `port` på servern på `iAddr`
`InputStream getInputStream()`	ger en ström för hämtning av data från den andra datorn
`OutPutStream getOutputStream()`	ger en ström för sändning av data till den andra datorn
`InetAddress getInetAddress()`	ger adressen som denna socket är ansluten till
`close()`	stänger förbindelsen

java.net.ServerSocket

Metoder (i urval)	Förklaring
`new ServerSocket(int port)`	ger en socket för lyssning på porten `port`
`Socket accept()`	väntar på att klient ansluter sig ger tillbaks klientens socket
`close()`	stänger förbindelsen

```
ServerSocket servSocket = new ServerSocket(1234);
while (true) {
    // wait for a client to connect
    Socket clientSocket = servSocket.accept();
    new clientHandler(clientSocket);
}
```

Exekverar klienthanteringen i en tråd

Exempel: Väderstation