F3
Java I/O - strömmar
Meddelandesändning med TCP

EDA095 Nätverksprogrammering

Roger Henriksson
Datavetenskap
Lunds universitet

Java I/O – Strömmar och filer

Ström (eng. Stream)

En ström är en sekvensiell följd av bytes (tecken).

In- och utmatning sker oftast i form av strömmar: Inmatning från tangentbordet, utskrift till ett terminalfönster.

Vi kan skriva bytes till strömmar och vi kan läsa bytes från strömmar.

Vi kan upprätta strömmar över ett nätverk: TCP

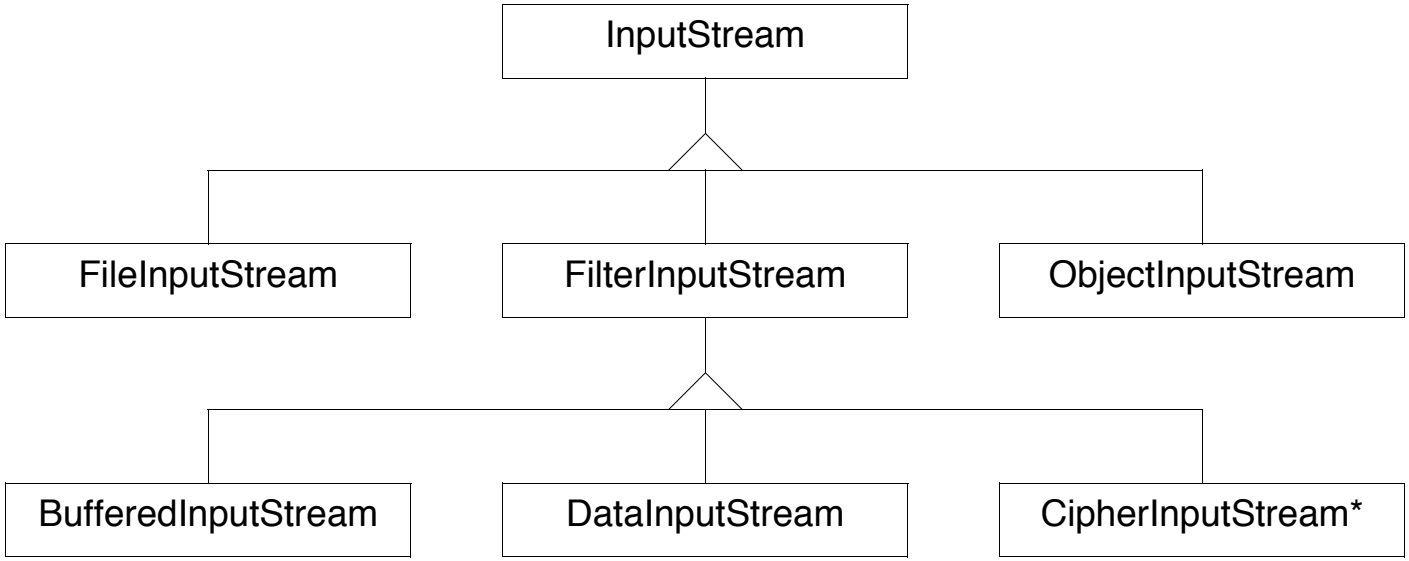
Fil (eng. File)

En fil är en ström som lagrats på ett sekundärminne.

Exempel: Era Javaprogram är lagrade i textfiler.

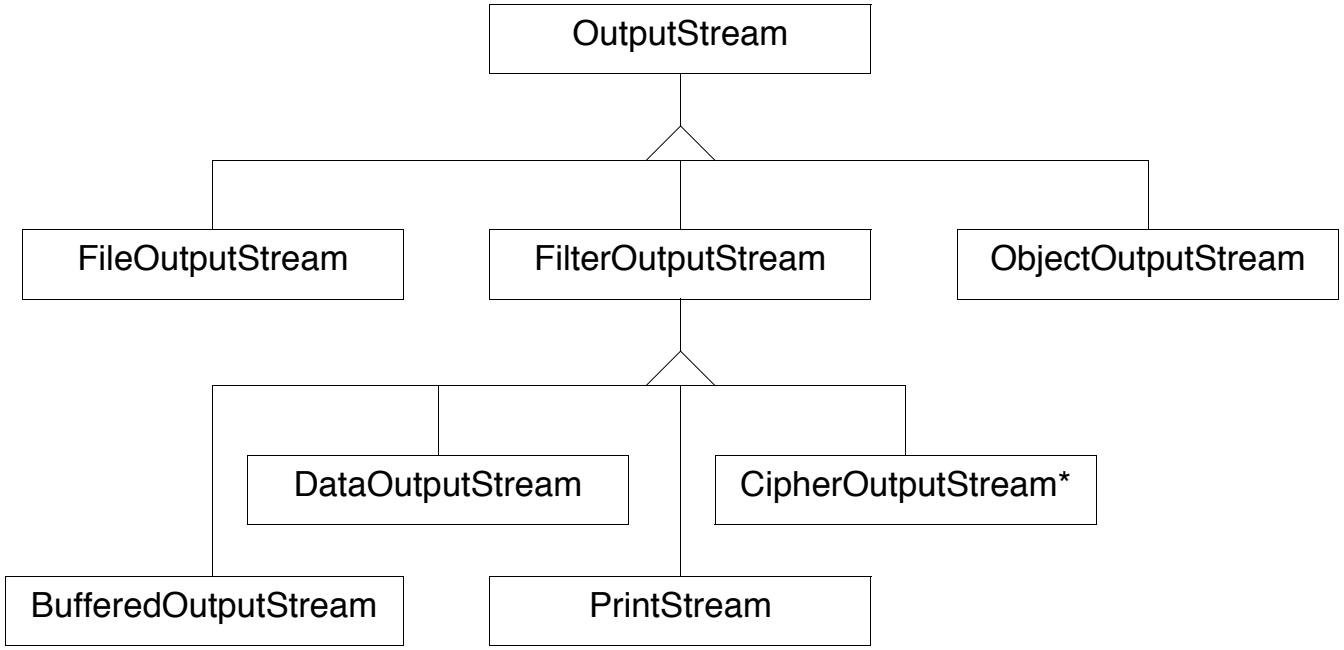
Strömmar i Java – inmatning

Klasserna för att hantera strömmar finns i paketet java.io.



Endast en delmängd...

Strömmar i Java – utmatning



Endast en delmängd...

InputStream

Abstrakt klass som representerar en inkommande ström.

Superklass till övriga InputStream-klasser.

Innehåller metoder för att läsa bytes.

```
public int abstract read() throws IOException;
public int read(byte[] input) throws IOException;
public int read(byte[] input,int offset,int length)
                                        throws IOException;

public long skip(long n) throws IOException;

public int available() throws IOException;

public void close() throws IOException;
```

OutputStream

Abstrakt klass som representerar en utgående ström.

Superklass till övriga OutputStream-klasser.

Innehåller metoder för att skriva bytes.

```
public abstract void write(int b) throws IOException;
public void write(byte[] data) throws IOException;
public void write(byte[] data, int offset, int length)
                                        throws IOException;

public void flush() throws IOException;

public void close() throws IOException;
```

Filter

Objekt av subklasserna till FilterInputStream och FilterOutputStream kan kopplas ihop (kedjas ihop) med InputStream- respektive OutputStream-objekt för att utöka deras funktionalitet.

Exempel:

Använd en BufferedInputStream för att öka prestanda vid inläsning från fil:

```
InputStream is = ...;
BufferedInputStream bis = new BufferedInputStream(is);
```

Motsvarande för utmatning:

```
OutputStream os = ...;
BufferedOutputStream = new BufferedOutputStream(os);
```

Buffring

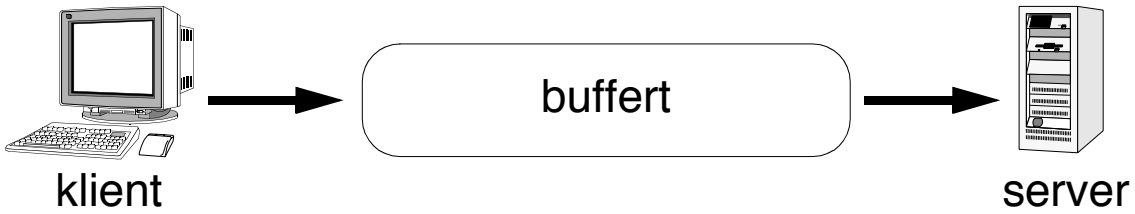
Buffring av strömmar ökar effektiviteten – högre genom-strömning per tidsenhet.

Ju mer data som sänds / tas emot åt gången ju färre under-liggande systemanrop / diskaccesser / nätverkspaket.

Klasserna BufferedInputStream / BufferedOutputStream implementerar buffring.

VARNING: Glöm inte att anropa flush()!

Risk för dödläge:



Meddelanden kan fastna i bufferten!

Data(Input/Output)Stream

Subklasser till FilterInputStream/FilterOutputStream.

Kan användas om man vill skriva annat än bytes på en ström. De konverterar mellan bytes och olika andra datatyper.

```
public void writeBoolean(boolean b);
public void writeByte(int b);
public void writeShort(int s);
public void writeChar(int c);
public void writeInt(int i);
public void writeLong(long l);
public void writeFloat(float f);
public void writeDouble(double d);
public void writeChars(String s);
public void writeBytes(String s);
public void writeUTF(String s);
```

```
public boolean readBoolean();
public byte readByte();
public char readChar();
public short readShort();
public int readInt();
public long readLong();
public float readFloat();
public double readDouble();
public String readUTF();
```

Alla operationer kan generera ett IOException.

UDP

Protokoll för att skicka korta meddelanden i form av paket.

- Begränsad paketstorlek – måste stycka upp det som ska överföras i flera mindre paket.
- Ingen garanti att paket kommer fram alls.
- Ingen garanti att paket kommer fram i rätt ordning.

Besvärligt att implementera mera komplicerade protokoll där en transaktion kräver flera meddelanden.

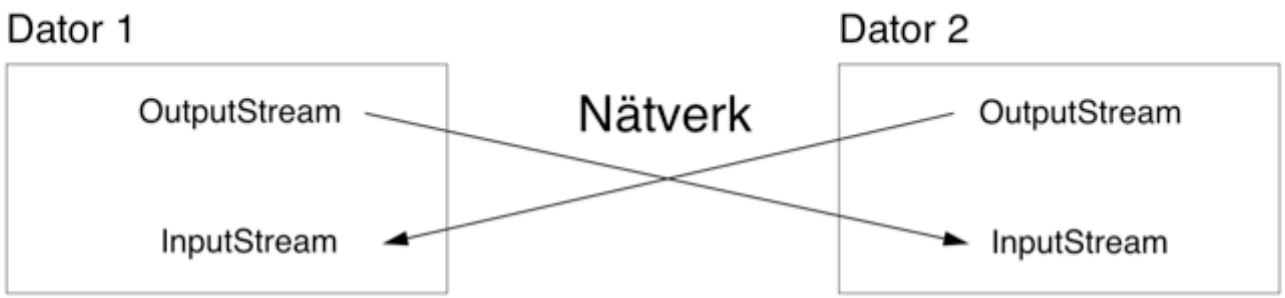
Okopplat (unconnected) protokoll

TCP

"Fast" uppkoppling mellan två portar på (vanligtvis) två olika datorer. Fast uppkoppling (connected protocol)!

- Automatisk omsändning / sortering av meddelanden.
- Ingen storleksbegränsning.

Ur applikationsprogrammets synvinkel fungerar en TCP-förbindelse som två strömmar (streams), en inkommande (InputStream) och en utgående (OutputStream):



Java och TCP

Socket

Motsvarar en upprättad förbindelse till vilken man kan skriva utgående bytes och läsa inkommande bytes.

ServerSocket

Används av en server för att vänta på uppkopplingar.

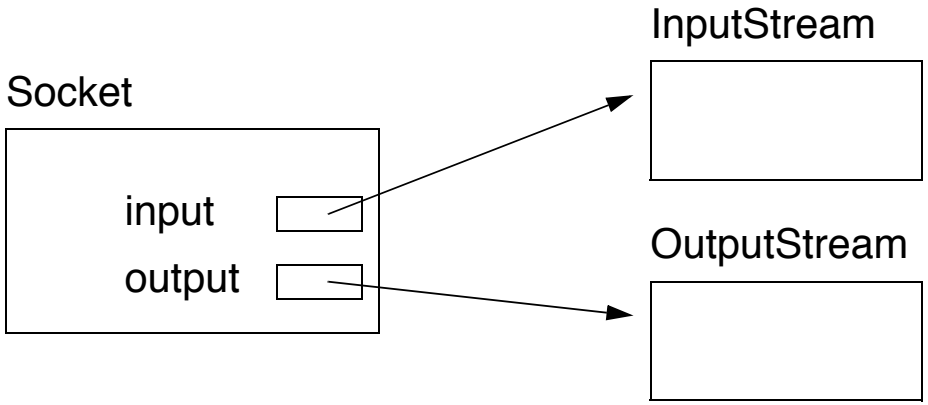
Stream I/O

InputStream och OutputStream från java.io.

Paketen java.net och java.io.

Socket – modell

Modell



Socket-objektet sköter övergripande egenskaper för uppkopplingen.

Stream-objekten har hand om in-/utmatning av bytes.

Stream-objekten skapas automatiskt.

Socket

Konstruktorer

```
public Socket(String host, int port)
                 throws UnknownHostException, IOException;
public Socket(InetAddress host,int port)
                 throws IOException;
```

Get-metoder

```
public InputStream getInputStream() throws IOException;
public OutputStream getOutputStream()throws IOException;
public InetAddress getInetAddress();
public int getPort();
public InetAddress getLocalAddress();
public int getLocalPort();
```

Socket, fortsättning

Inställningar

```
public void setTcpNoDelay(boolean on)
                                throws SocketException;
public boolean getTcpNoDelay() throws SocketException;
public void setSoTimeout(int milliseconds)
                                throws SocketException;
public int getSoTimeout() throws SocketException;
...
```

Koppla ned

```
public void close() throws IOException;
```

Skapa en socket (klient)

```
Socket socket = null;
InputStream input = null;
OutputStream output = null;
try {
   socket = new Socket("www.cs.lth.se",80);
   input = socket.getInputStream();
   output = socket.getOutputStream();
} catch (UnknownHostException e) {
   System.out.println(e);
   System.exit(1);
} catch (IOException e) {
   System.out.println(e);
   System.exit(1);
}
```

DEMO - SimpleTelnet

Att läsa stora block

Att läsa en byte i taget med read() är mycket ineffektivt.
Lösning:

```
public int read(byte[] input,int offset,int length)
                                        throws IOException;
```

Varning! Ingen garanti att angivet antal bytes läses!
Exempel: Vi vill läsa exakt 100 byte.

```
byte[] buffer = new byte[100];
int read = 0;
int result = 0;
while (read<100 && result!=-1) {
   result = input.read(buffer,read,100-read);
   if (result!=-1)
      read = read+result;
}
```

TCP-server – skelett

```
while (true) {
    receive(client, command, parameters);
    switch (command) {
        case commandA:
            result = doCommandA(parameters);
            break;
        case commandB:
            result = doCommandB(parameters);
            break;
        case commandC:
            result = doCommandC(parameters);
            break;
        ...
        default: ...
    }
    send(client, result);
}
```

ServerSocket

Fungerar som en telefonist - tar emot uppkopplingar på serversidan och skapar ett motsvarande Socket-objekt.

Klienten och servern kommunicerar via denna Socket.

En servers livscykel

1. Skapa en ServerSocket.
2. Vänta på uppkoppling (accept()). Socket skapas.
3. Hämta strömmar med getInputStream() respektive getOutputStream().
4. Servern och klienten kommunicerar med varandra.
5. Förbindelsen kopplas ned.
6. Gå till steg 2 och vänta på ny uppkoppling.

ServerSocket, fortsättning

Konstruktorer

```
public ServerSocket(int port) throws IOException;
public ServerSocket(int port, int queueLength)
                                    throws IOException;
...
```

Vänta på en uppkoppling

```
public Socket accept() throws IOException;
...
```

Koppla ned servern

```
public void close() throws IOException;
```

ServerSocket, fortsättning

Get-metoder

```
public InetAddress getInetAddress();
public int getLocalPort();
public int getSoTimeout() throws IOException;
```

Set-metoder

```
public void setSoTimeout(int timeout)throws IOException;
```

DEMO - ToUpperServer

Skicka/ta emot tecken

Ibland vill man skicka tecken istället för bytes och vill slippa att själv göra omvandlingen i sitt program.

Reader/Writer - en parallell klasshierarki till InputStream/ OutputStream som hanterar tecken enligt en given tecken-kodning.

Vi gör om strömmar till en reader/writer genom att kapsla in den (precis som med FilterInput/OutputStream):

```
OutputStreamWriter out = new
OutputStreamWriter(socket.getOutputStream());
InputStreamReader in = new
                InputStreamReader(socket.getInputStream());
```

Se kursboken, kapitel 4, och Javas klassdokumentation!

Skicka/ta emot strängar

Klasserna BufferedReader och PrintWriter representerar sträng- och textorienterade strömmar.
De är buffrade och hanterar rader av tecken effektivt via metoden `readLine()`:

```
InputStream is = ...
OutputStream os = ...
BufferedReader in = new
              BufferedReader(new InputStreamReader(is));
PrintWriter out = new PrintWriter(os,true);
```

Se kursboken, kapitel 4, och Javas klassdokumentation!

Hantera flera TCP-anslutningar

Vad händer om en andra klient kopplar upp sig medan servern är upptagen med att betjäna den första klienten?

Blockering eller förvägrad uppkoppling!

DEMO - ToUpperServer

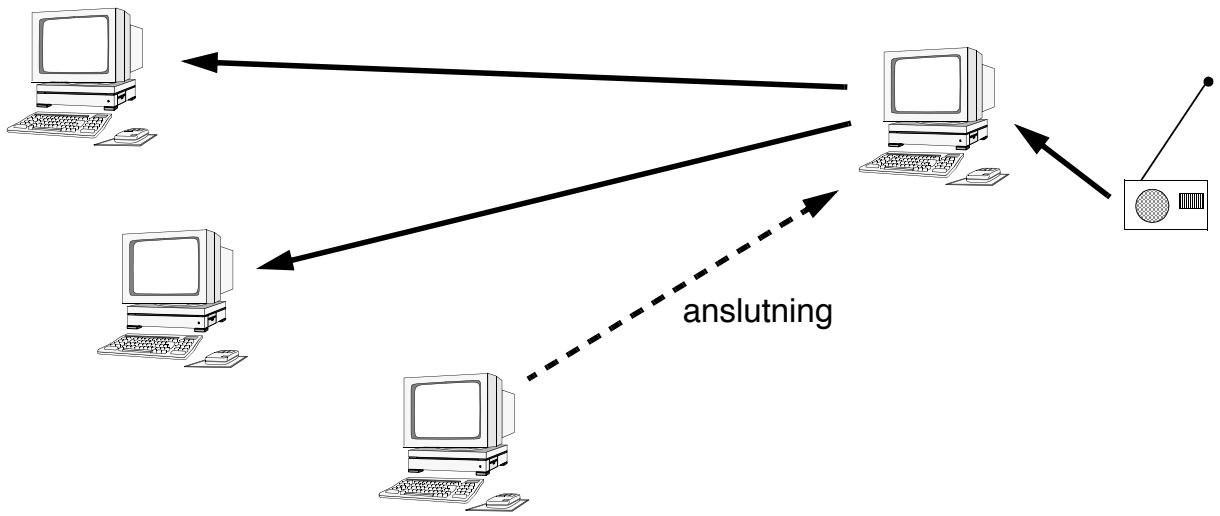
Visst hade det varit bra om ert serverprogram kunde göra flera saker samtidigt? Tex betjäna flera olika klienter?

Lösningar:

- Trådar...
- "Non-blocking I/O" - java.nio, java.nio.channels
  Introducerat i Java 1.4.

Icke-blockerande I/O i C

Internetradio anno 1992 på Datavetenskap.



- raclient.c
- raserver.c

Icke-blockerande I/O i Java

Nya Javaklasser i paketen java.nio och java.nio.channels:

- InetSocketAddress
- ServerSocketChannel
- SocketChannel
- ByteBuffer
- Selector
- SelectorKey

Se kursboken, kapitel 12.

Exempel: Echo server

Java Network Programming (3rd ed.) sid 407-408

DEMO – EchoServer