Notater til OOPVA60

Thomas Nordli <tn@hive.no>

10. juni 2010

Kompendium 1 / 2010 Høgskolen i Vestfold
Copyright: Høgskolen i Vestfold / Thomas Nordli
Issn 0808-1328

Innhold

I **7**

1 TCP/IP, Sockets, klient/tjener **9**
1.1 Internett . . . 9
1.2 Transmission Control Protocol . . . 11
1.3 User Datagram Protocol . . . 12
1.4 Sockets . . . 12
1.5 Øvingsoppgaver . . . 18

2 Threads, synkronisering **19**
2.1 Løsningsforslag på forrige ukes øving . . . 19
2.2 Tråder . . . 21
2.3 RaceCondition . . . 23
2.4 Øvelsesoppgave . . . 27

3 Ikke-blokkerende-IO, URL-er, filer, kommandolinjeargumenter **29**
3.1 Løsningsforslag på forrige øvelsesoppgave . . . 29
3.2 Mer trådsynkronisering . . . 30
3.3 Ikke-blokkerende tjener . . . 32
3.4 URL . . . 35
3.5 Lesing av fil . . . 35
3.6 Kommandolinje-argumenter . . . 36
3.7 Øvelsesoppgaver . . . 37

4 RandomAccess-filtilgang, serialisering, vector **39**
4.1 Løsningsforslag på forrige øvelsesoppgave . . . 39
4.2 RandomAccessFile . . . 42
4.3 Serializable . . . 44
4.4 Vector . . . 45
4.5 Øvelsesoppgaver . . . 46

5 HTML,CGI **47**
5.1 Løsningsforslag til oppgaver fra forrige gang . . . 47
5.2 Grunnlenggende HTML . . . 49
5.3 Grunnleggende CGI . . . 50
5.4 Øvelser . . . 54

6 Oppsummering del 1 **55**
6.1 Løsningsforslag til oppgave fra forrige gang . . . 55
6.2 IP-adresse . . . 56

6.3 Sockets . . . 58
6.4 Threads, synkronisering . . . 61
6.5 Ikke-blokkerende-IO . . . 64
6.6 filer, URL-er kommandolinje, argumenter . . . 67
6.7 RandomAccess-filtilgang, serialisering, vector . . . 68
6.8 HTML, CGi . . . 69
6.9 Øvelser . . . 70

II **73**

7 RMI **75**
7.1 Tradisjonelt to kommunikasjonsformer: . . . 75
7.2 Problem: Kan ikke lese minnet på fjern maskin. To måter å løse problemet på: . . . 75
7.3 Remote Method Invocation . . . 75
7.4 Implementasjon av tjener og klient . . . 76
7.5 Øvelsesoppgave 07 . . . 79

8 RMI II **81**
8.1 Løsningsforslag fra forrige øvelse . . . 81
8.2 RMI-Security . . . 84
8.3 Obligatiorisk oppgave . . . 86

9 Servlets **87**
9.1 Om servlets . . . 87
9.2 Eksempler . . . 87
9.3 Øvelsesoppgaver . . . 91

10 JavaServer Pages **93**
10.1 Hva er JSP? . . . 93
10.2 Hvorfor JSP ble introdusert? . . . 93
10.3 Når brukes JSP, og når brukes servlets? . . . 94
10.4 kompilering/kjøring . . . 95
10.5 JElementer . . . 96
10.6 Implisitte JSP-objekter . . . 98
10.7 Tilgang til DB via JDBC fra JSP . . . 99
10.8 To hovedmetoder for samarbeid mellom servlets og JSP . . . 100
10.9 Error pages . . . 102
10.10 Øvelse . . . 103

11 JavaBeans **107**
11.1 Løsning på tidligere oppgave . . . 107
11.2 JavaBeans . . . 108
11.3 JavaBeans i JSP . . . 109
11.4 Eksempler . . . 111
11.5 Øvelse . . . 116

12 Applets og CORBA **117**
12.1 Applets . . . 117
12.2 CORBA . . . 126
12.3 Øvelser . . . 127

13 Oppsummering del 2 **129**
13.1 Remote Method Invocation . . . 129
13.2 Servlets . . . 133
13.3 JavaServer Pages . . . 135
13.4 JavaBeans . . . 141
13.5 Applets . . . 144
13.6 Oppgave-eksempler . . . 149

14 Eksamen våren 2009 **151**
14.1 Oppgave 1 (25%) . . . 151
14.1.1 a) . . . 151
14.1.2 Løsningsforslag . . . 151
14.1.3 b) . . . 151
14.1.4 Løsningsforslag . . . 151
14.1.5 c) . . . 151
14.1.6 Løsningsforslag . . . 151
14.1.7 d) . . . 152
14.1.8 Løsningsforslag . . . 152
14.1.9 e) . . . 152
14.1.10 Løsningsforslag . . . 152
14.2 Oppgave 2 (25%) . . . 152
14.2.1 a) . . . 152
14.2.2 Løsningsforslag . . . 152
14.2.3 b) . . . 152
14.2.4 Løsningsforslag . . . 152
14.2.5 c) . . . 152
14.2.6 Løsningsforslag . . . 153
14.2.7 d) . . . 153
14.2.8 Løsningsforslag . . . 153
14.2.9 e) . . . 153
14.2.10 Løsningsforslag . . . 153
14.3 Oppgave 3 (25%) . . . 153
14.3.1 a) . . . 154
14.3.2 Løsningsforslag . . . 154
14.3.3 b) . . . 154
14.3.4 Løsningsforslag . . . 154
14.3.4.1 10 . . . 154
14.3.4.2 12 . . . 154
14.3.4.3 13 . . . 154
14.3.4.4 14 . . . 154
14.3.4.5 16 . . . 154
14.3.4.6 17 . . . 155
14.3.4.7 18 . . . 155
14.3.4.8 19 . . . 155

14.3.4.9 20 . . . 155
14.3.5 c) . . . 155
14.3.6 Løsningsforslag . . . 155
14.3.7 d) . . . 155
14.3.8 Løsningsforslag . . . 155
14.3.9 e) . . . 156
14.3.10 Løsningsforslag . . . 156
14.4 Oppgave 4 (25%) . . . 156
14.4.1 a) . . . 157
14.4.2 b) . . . 157
14.4.3 Løsningsforslag . . . 157
14.4.4 c) . . . 157
14.4.5 Løsningsforslag . . . 157
14.4.6 d) . . . 158
14.4.7 Løsningsforslag . . . 158
14.4.8 e) . . . 158
14.4.9 Løsningsforslag . . . 158

Om skriftet

Skriftet *Notater til OOPVA60* er et kompendium som inneholder undervisningsmateriale. Undervisningsmaterialet er produsert i forbindelse med undervisning i faget *Objektorientert programmering II.* , våren 2010. Faget ble tilbudt som valgfag, for ingeniørstudenter med *grunnleggende ferdigheter objektorientert programmering.*

Kompendiet er en sammenstilling av:

- notater
- eksempler
- oppgaver
- løsningsforslag

, som ble brukt i forbindelse med undervisningen.

Om forfatteren

Forfatteren, Thomas Nordli, var fagansvarlig for kurset skoleåret 2009/2010. Han er ansatt som høgskolelektor ved *Fakultet for teknologi og maritime fag* ved *Høgskolen i Vestfold*, hvor han underviser i *Operativsystemer*, *Databaser* og *Programmering.*

Del I

Kapittel 1

TCP/IP, Sockets, klient/tjener

1.1 Internett

- sammenkobling av ulike nett
- maskiner kommuniserer med Internet-Protokollen (IP)
- Rutere formidler trafikk mellom nettene
- Pakkeswitchet
- Best effort

lagdelt modell

- applikasjonslag
- transportlag
- (inter)nettverkslag
- forbindelse (link)
- (fysisk)

Hver node/vert har sin egen adresse

IPv4-adresse: 32 bit

- dot quad: fire 8-bits nummer
- eks.: 128.39.105.10
- IPv6-adresse: 128 bit
- De numeriske IP-adressene er komplementert med alfabetiske domenenavn v.h.a. domenenavn-systemet DNS.

Eksempel

- Klassen MittNavnEr demonstrerer hvordan bruken av klassen java.net.InetAddress kan brukes til å hente informasjon fra navnetjenesten i verts-operativsystemet.

Listing 1.1: eksempler/MittNavnEr.java

```
1  public class MittNavnEr {
2
3    public static void main(String[] args)
4
5    /*
6     * Unntaket java.net.UnknownHostException kastes ved mislykket
7     * navneoppslag.
8     *
9     * Med navneoppslag menes å fremskaffe ip-adressen til vert basert
10    * på vertens domenenavn.
11    *
12    * Både java.net.InetAddress.getLocalHost() og
13    * java.net.InetAddress.getByName(maskin) som benyttes i koden under
14    * kan kaste dette unntaket.
15    */
16
17   throws java.net.UnknownHostException {
18
19     java.util.Scanner inngang = new java.util.Scanner(System.in);
20
21     /*
22      * For å opprette et objekt av klassen java.net.InetAddress som
23      * refererer til "localhost" kan den statiske metoden
24      * java.net.InetAddress.getLocalHost() benyttes.
25      */
26
27     java.net.InetAddress adresse = java.net.InetAddress.getLocalHost();
28
29     System.out.println(
30         "Hallo, mitt navn er " +
31         adresse +
32         ". Skriv navnet på en annen maskin:"
33         );
34
35     String maskin = inngang.next();
36
37     /*
38      * For å gjøre et navneoppslag kan den statiske metoden
39      * java.net.InetAddress.getByName(String vert) benyttes.
40      *
41      * Dersom argumentet inneholder et domenenavn,
42      * vil metoden be vertsoperativsystemet gjøre et navneoppslag.
43      * (i DNS eller filen hosts).
44      *
45      *  Dersom 'vert' inneholder en tekststreng med ip-adresse,
46      *  gjøres kun en sjekk på om det er på riktig format.
47      */
48
49     adresse = java.net.InetAddress.getByName(maskin);
50
51     System.out.println(
```

```
52            "Adressen til" +
53            maskin +
54            " er " +
55            adresse +
56            "."
57            );
58    }
59 }
```

1.2 Transmission Control Protocol

- På transportlaget
- Forbindelsesorientert
- Tilbyr pålitelig overføring til applikajsonslaget
- Applikasjonenen adresseres med portnummer

tre faser

1. oppkobling

treveis håndtrykk

- Klient: SYN
- Tjener: SYN/ACK
- Klient: ACK

2. dataoverføring

- sjekksummer
- sekvensnumre
- kvitteringer (ACK)
- flytkontroll
- overbelastningskontroll (congestion control)

3.nedkobling

fireveis håndtrykk

- hver ende kobler ned med FIN
- besvares med ACK

1.3 User Datagram Protocol

- På transportlaget
- Forbindelsesløs
- Applikasjonsmultiplexing (m/portnumre)
- Valgfri sjekksum
- Ingen garantier
- Ingen tilstandsinfo lagres hos avsender
- Raskere enn TCP

1.4 Sockets

- Når to prosesser kommuniserer over et et nettverk basert på internett-protokollen (IP), brukes vanligvis en toveis kommunikasjonskanal. Endepunktene for en slik kanal kalles sockets. Sockets defineres av ipadresse og port-nummer for hver ende av kanalen.

TCP - klient/tjener i java

TCP-tjener

1. Opprette 'ServerSocket'

- ServerSocket serverSocket = new ServerSocket(PORTNUMMER);

2. Sette tjener i vente-tilstand

- Socket socket = serverSocket.accept();

3. Sette opp inn- og ut- strømmer

- socket.getInputStream();
- socket.getOutputStream();

4. Sende og motta data

- I henhold til applikasjons-protokoll

5. Stenge forbindelsen

- socket.close();

Eksempler

TCPSensor: Å lytte på en TCP-port

- Klassen TCPSensor demonstrerer hvordan et java-program kan settes opp til å lytte på en TCP-port.

Listing 1.2: eksempler/TCPSensor.java

```
1
2  import java.io.IOException;
3  import java.net.ServerSocket;
4  import java.net.Socket;
5
6
7  public class TCPSensor {
8
9    public static void main(String[] args) throws IOException {
10
11     ServerSocket tjenerSocket = new ServerSocket(8888);
12     tjenerSocket.setReuseAddress(true);
13
14     System.out.println(
15         "Tjeners adresse: \t"       + tjenerSocket.getInetAddress()    +
16         "\nSensor aktivert på port\t" + tjenerSocket.getLocalPort() );
17
18     Socket forbindelse = tjenerSocket.accept();
19
20     System.out.println(
21         "Forbindelsens lokale adresse:\t" + forbindelse.getLocalAddress() +
22         "\nForbindelsens lokale port:\t"       + forbindelse.getLocalPort()
               +
23         "\nForbindelsens fjerne adresse:\t" + forbindelse.getInetAddress()
               +
24         "\nForbindelsens fjerne port"       + forbindelse.getPort() );
25
26     tjenerSocket.close();
27     forbindelse.close();
28
29   }
30
31 }
```

For å teste programmet:

- Start programmet med kommandoen java TCPSensor i et konsoll-vindu
- Koble deg til TCP-port 8888 fra en annet program (f.eks. netcat fra et annet konsoll-vindu).

TCP-tjener

- Klassen TCPsensorTjener kjører en evig løkke, for kontinuerlig å vente på tcp-oppkoblinger mot port 8888. Responsen blir nå sendt tilbake til klienten, og ikke til standard utgang, slik som i forrige eksempel. Test f.eks. med netcat: nc localhost 8888.

Listing 1.3: eksempler/TCPSensorTjener.java

```
1
2  import java.io.IOException;
3  import java.io.OutputStream;
4  import java.io.PrintWriter;
5  import java.net.ServerSocket;
6  import java.net.Socket;
7
8  public class TCPSensorTjener {
9
10   public static void main(String[] args) throws IOException {
11
12     ServerSocket tjenerSocket = new ServerSocket(8888);
13     tjenerSocket.setReuseAddress(true);
14     PrintWriter utskriver;
15     OutputStream utStrom;
16     Socket forbindelse;
17
18     while (true) {
19
20       forbindelse = tjenerSocket.accept();
21       utStrom = forbindelse.getOutputStream();
22       utskriver = new PrintWriter(utStrom);
23
24       utskriver.format(
25           "Din oppkobling fra %s er detektert!\n",
26           forbindelse.getRemoteSocketAddress() );
27
28       utskriver.flush();
29       forbindelse.close();
30
31     }
32
33   }
34
35 }
```

Kontinuerlig kommunikasjon

- Klassen TCPSkravler_1 opprettholder forbindelsen fra klienten. Det klienten sender, returneres. Tjeneren kan kun betjene en klient om gangen.TCPSkravler_1.java

Listing 1.4: eksempler/TCPSkravler_1.java

```
1  public class TCPSkravler_1 {
2
3    public static void main(String[] args)
4    throws java.io.IOException {
5
6      java.net.ServerSocket tjenerSocket = new java.net.ServerSocket(8888);
7      tjenerSocket.setReuseAddress(true);
8      java.io.PrintWriter utskriver;
9      java.io.InputStream innStrom;
10     java.net.Socket forbindelse;
11     java.util.Scanner innleser;
12
13     while (true) {
```

```
14
15          forbindelse = tjenerSocket.accept();
16
17          utskriver = new java.io.PrintWriter(forbindelse.getOutputStream());
18          innStrom = forbindelse.getInputStream();
19          innleser = new java.util.Scanner(innStrom);
20
21          while(true) {
22            utskriver.format(
23                "Interessant at du sier \"%s\".\n",
24                innleser.nextLine() );
25
26            utskriver.flush();
27          }
28        }
29      }
30  }
```

TCP-klient

1. Opprette forbindelse til tjener

- Socket socket = new Socket(INETADDR, PORT);

2. Sette opp inn- og ut- strømmer

- socket.getInputStream();
- socket.getOutputStream();

4. Sende og motta data

- I henhold til applikasjons-protokoll

5. Stenge forbindelsen

- socket.close();

UDP - klient/tjener

UDP-tjener

1. Opprette 'DatagramSocket'-objekt

- DatagramSocket datagramSocket = new DatagramSocket(PORT);

2. Opprette byte-buffer

- byte[] buffer = new byte[BUFFERSTR];

3. Opprette 'DatagramPacket'-objekt

- DatagramPacket innPakke = new DatagramPacket(buffer, buffer.length);

4. Motta pakke

- datagramSocket.recieve(innPakke);

5. Finne avsenders adresse og portnr.

- InetAddress adr = innPakke.getAddress();
- int port = innPakke.getPort();

6. Hente ut pakkens data

- byte[] inndata = innPakke.getData();

7. Opprette datagram (for respons)

- DatagramPacket utPakke = new DatagramPacket(utData, utData.length(), adr, port);

8. Send datagram

- datagramSocket.send(utPakke);

9. Stenge 'DatagramSocket'

- datagramSocket.close();

Eksempel: Å lytte på en UDP-port

- Klassen UDPSensor tar i mot datagram på en udp-port. Legg merke til at det ikke opprettes en ny sockets i forbindelse med mottaket.
- For å teste UDP-sensoren, kan du bruke netcat med opsjonen -u (for udp).
- UDPSensor.java

Listing 1.5: eksempler/UDPSensor.java

```
1  public class UDPSensor {
2
3    public static void main(String[] args)
4    throws java.io.IOException {
5
6      java.net.DatagramSocket datagramSocket = new java.net.DatagramSocket
           (8888);
7      datagramSocket.setReuseAddress(true);
8
9      byte[] buffer = new byte[256];
10     java.net.DatagramPacket datagram = new java.net.DatagramPacket(buffer,
           buffer.length);
11
12     System.out.println(
13         "Tjeners adresse:\t"           + datagramSocket.getLocalAddress() +
14         "\nSensor aktivert på port\t" + datagramSocket.getLocalPort() );
```

```
15
16      datagramSocket.receive(datagram);
17
18      System.out.println(
19          "Datagrammet fra porten "    + datagram.getPort() +
20          " på maskinen "              + datagram.getAddress() +
21          " er detektert.");
22
23      datagramSocket.close();
24    }
25  }
```

UDP-klient

1. Opprette 'DatagramSocket'-objekt

- DatagramSocket datagramSocket = new DatagramSocket();

2. Opprette datagram

- DatagramPacket utPakke = new DatagramPacket(utData, utData.length(), adr, port);

3. Send datagram

- datagramSocket.send(utPakke);

4. Opprette byte-buffer

- byte[] buffer = new byte[BUFFERSTR];

5. Opprett (inn-)'DatagramPacket'-objekt

- DatagramPacket innPakke = new DatagramPacket(buffer, buffer.length);

6. Motta datagram

- datagramSocket.recieve(innPakke);

7. Hente ut data fra buffer

- byte[] inndata = innPakke.getData();

8. Stenge 'DatagramSocket'

- datagramSocket.close();

- Implementasjon av klientprogram er øvingsoppgave.

1.5 Øvingsoppgaver

Oppgave 1.1

- Logg inn på matiksi.hive.no.
- Start eclipse.
- Lage et "halloverden"-program.
- Kjør "halloverden"-programmet fra kommandolinjen.

Oppgave 1.2

- Programmer en klient til eksempelprogrammet TCPSkravler_1.

Oppgave 1.3

- Programmer en klient til eksempelprogrammet UDPSensor

Kapittel 2

Threads, synkronisering

2.1 Løsningsforslag på forrige ukes øving

Bugfiks av TCPSkravler

- Bytte ut "while (true)" med "while (innleser.hasNextLine())"
- TCPSkravler_1v2.java

Listing 2.1: losningsforslag/TCPSkravler_1v2.java

```
1
2  import java.net.ServerSocket;
3  import java.io.IOException;
4  import java.io.InputStream;
5  import java.io.PrintWriter;
6  import java.util.Scanner;
7  import java.net.Socket;
8
9  public class TCPSkravler_1v2 {
10
11   public static void main(String[] args) throws IOException {
12
13     ServerSocket tjenerSocket = new ServerSocket(8888);
14     tjenerSocket.setReuseAddress(true);
15     PrintWriter utskriver;
16     InputStream innStrom;
17     Socket forbindelse;
18     Scanner innleser;
19
20     while (true) {
21
22       forbindelse = tjenerSocket.accept();
23
24       utskriver = new PrintWriter(forbindelse.getOutputStream());
25       innStrom = forbindelse.getInputStream();
26       innleser = new Scanner(innStrom);
27
28       while(innleser.hasNextLine()) {
29         utskriver.format(
30             "Interessant at du sier \"%s\".\n",
31             innleser.nextLine() );
32
```

```
33            utskriver.flush();
34
35          }
36
37          forbindelse.close();
38
39        }
40
41    }
42
43 }
```

- TCPSkraver_1v2 kan kun håndtere en klient av gangen. Hva kan vi gjøre med det?

Oppgave 1.2

- Programmer en klient til eksempelprogrammet TCPSkravler_1.
- TCPKlient.java

Listing 2.2: losningsforslag/TCPKlient.java

```
1  import java.io.IOException;
2  import java.io.PrintWriter;
3  import java.net.InetAddress;
4  import java.net.Socket;
5  import java.net.UnknownHostException;
6  import java.util.Scanner;
7
8
9  public class TCPKlient {
10
11   public static void main(String[] args) throws UnknownHostException,
         IOException {
12
13     Scanner stdInn, socketInn;
14     PrintWriter socketUt;
15     Socket s;
16
17     s    = new Socket( InetAddress.getByName("localhost"), 8888);
18     socketInn = new Scanner(s.getInputStream());
19     socketUt  = new PrintWriter(s.getOutputStream());
20     stdInn = new Scanner(System.in);
21
22     while (stdInn.hasNextLine()){
23
24       socketUt.println(stdInn.nextLine());
25       socketUt.flush();
26
27       System.out.println(socketInn.nextLine());
28
29     }
30
31     s.close();
32
33   }
34 }
```

Oppgave 1.3

- Programmer en klient til eksempelprogrammet UDPSensor
- UDPKlient.java

Listing 2.3: losningsforslag/UDPKlient.java

```
1  import java.io.IOException;
2  import java.net.DatagramPacket;
3  import java.net.DatagramSocket;
4  import java.net.InetAddress;
5
6
7  public class UDPKlient {
8
9
10   public static void main(String[] args) throws IOException {
11
12     DatagramSocket s;
13     DatagramPacket d;
14
15     s = new DatagramSocket();
16     d = new DatagramPacket("test".getBytes(),  4, InetAddress.getLocalHost
           (), 8888);
17
18     s.send(d);
19     s.close();
20     }
21 }
```

2.2 Tråder

- En sekvens av kommandoer som utføres kalles en tråd
- En prosess (java vm) kan håndtere flere tråder på en gang
- Det er alltid minst en tråd om kjører, når et javaprogram kjøres
- Når main() kalles, starter en tråd, som "drepes" når main() terminerer.
- Raskere å håndtere tråder enn prosesser
- Trådene deler minneområdet. De deler globale variabler
- Hver tråd har i tillegg eget minneormådet.
- Et java-objekt kan kjøres i en egen tråd, dersom det implementerer grensesnittet (interfacet) *Runnable*.
- Dette gjøres på to måter:
- 1. lage en klasse som utvider (extends) klassen Thread
- 2. instansiere et Thread-objekt med et objekt som implementerer grensesnittet (interface) Runnable som argument.

Eksempler på bruk av tråder i java

extends Thread

- Traad_1.java

Listing 2.4: eksempler/Traad_1.java

```
1
2  import java.io.IOException;
3
4  public class Traad_1 extends Thread {
5
6    public static void main(String[] args) throws IOException {
7
8      Traad_1 t1, t2;
9
10     t1 = new Traad_1();
11     t2 = new Traad_1();
12
13     t1.start();
14     t2.start();
15   }
16
17   public void run(){
18
19     int i;
20     for (i=0;i<5;i++)
21       System.out.println(this.getName());
22   }
23 }
```

- Traad_2.java

Listing 2.5: eksempler/Traad_2.java

```
1  import java.io.IOException;
2
3
4  public class Traad_2 extends Thread {
5    public static void main(String[] args) throws IOException {
6
7      int i;
8      for (i=0; i<5; i++)
9        new Traad_2().start();
10   }
11
12   public void run(){
13
14     int i;
15     for (i=0;i<5;i++) {
16       System.out.println(this.getName());
17       try {
18         sleep( (int)(Math.random()*1000) );
19
20       } catch (InterruptedException e) {
21         e.printStackTrace();
22       }
```

```
23      }
24    }
25  }
```

implements Runnable

- Traad_3.java

Listing 2.6: eksempler/Traad_3.java

```
1
2  public class Traad_3 implements Runnable{
3
4    public static void main(String[] args) {
5
6      new Traad_3();
7    }
8
9    public Traad_3 () {
10     int i;
11     for (i=0; i<5; i++)
12       new Thread(this).start();
13   }
14
15   public void run(){
16
17     int i;
18     for (i=0;i<5;i++) {
19       System.out.println(Thread.currentThread().getName());
20       try {
21         Thread.sleep( (int)(Math.random()*1000) );
22
23       } catch (InterruptedException e) {
24         e.printStackTrace();
25       }
26     }
27   }
28 }
```

2.3 RaceCondition

Eksempler

- Balanse_1.java

Listing 2.7: eksempler/Balanse_1.java

```
1
2  import java.util.Random;
3
4  public class Balanse_1 extends Thread{
5
6    static final int T = 1000; // Totalsummen i spillet
7    static final int N = 5;    // Antall konti
8    static private int konto[] = new int[N];
```

```
9
10    Random slumptall = new Random();
11
12    public static void main(String[] args)  {
13
14      opprettKonti();
15      new Balanse_1().start();
16
17      while(true){
18
19        try {
20          sleep(1000);
21        } catch (InterruptedException e) {
22          e.printStackTrace();
23        }
24
25        skriv_saldo();
26
27      }
28    }
29
30    public void run()  {
31
32      int til, fra, belop;
33
34      while(true){
35
36        fra=Math.abs( slumptall.nextInt() ) %N;
37        til=Math.abs( slumptall.nextInt() ) %N;
38        belop=Math.abs( slumptall.nextInt() ) % ( (konto[fra]/10) );
39
40        konto[fra]-=belop;
41        konto[til]+=belop;
42
43      }
44    }
45
46    private static void opprettKonti(){
47
48      int i;
49      for (i=0;i<N;i++)
50        konto[i]=T/N;
51
52      skriv_saldo();
53    }
54
55    private static void skriv_saldo(){
56
57      int sum,i;
58
59      for (i=0, sum=0; i<N; i++){
60
61        System.out.printf("konto_%d:\t%d\n", i, konto[i]);
62        sum+=konto[i];
63      }
64
65      System.out.printf("----------\nTotalt:\t%d\n==========\n",sum);
66
```

```
67    }
68  }
```

- Balanse_2.java

Listing 2.8: eksempler/Balanse_2.java

```
1
2  import java.util.Random;
3
4
5  public class Balanse_2 extends Thread{
6
7    static final int T = 1000; // Totalsummen i spillet
8    static final int S = 10;   // Antall spekulanter som flytter penger
9    static final int N = 5;    // Antall konti
10
11   static int konto[] = new int[N];
12
13   static Random slumptall = new Random();
14
15   public static void main(String[] args) throws InterruptedException  {
16
17     opprettKonti();
18     opprettSpekulanter();
19
20     while (true){
21       skriv_saldo();
22       sleep(1000);
23     }
24   }
25
26   private static void opprettSpekulanter(){
27
28      int i;
29      for (i=0; i<S; i++)
30        new Balanse_2().start();
31   }
32
33   public void run()  {
34
35     int til, fra, belop;
36
37     while(true){
38
39       fra=Math.abs( slumptall.nextInt() ) %N;
40       til=Math.abs( slumptall.nextInt() ) %N;
41       belop=Math.abs( slumptall.nextInt() ) % ( (konto[fra]/10) );
42
43       konto[fra]-=belop;
44       konto[til]+=belop;
45
46     }
47
48   }
49
50   private static void opprettKonti() throws InterruptedException{
51
```

```
52      int i;
53
54      for (i=0;i<N;i++)
55        konto[i]=T/N;
56
57      skriv_saldo();
58
59    }
60
61    private static void skriv_saldo() throws InterruptedException{
62
63      int sum, i;
64
65      for (i=0, sum=0; i<N; i++){
66        System.out.printf("konto %d:\t%d\n", i, konto[i]);
67        sum+=konto[i];
68      }
69
70      System.out.printf("--------------\nTotalt:\t%d\n==============\n",sum);
71
72    }
73  }
```

- Balanse_3.java

Listing 2.9: eksempler/Balanse_3.java

```
1  public class Balanse_3 extends Thread{
2
3    static final int T = 1000; // Totalsummen i spillet
4    static final int S = 10;   // Antall spekulanter som flytter penger
5    static final int N = 5;    // Antall konti
6
7    static int konto[] = new int[N];
8
9    static java.util.Random slumptall = new java.util.Random();
10
11   public static void main(String[] args) throws InterruptedException  {
12
13     opprettKonti();
14     opprettSpekulanter();
15
16     while (true){
17       skriv_saldo();
18       sleep(1000);
19     }
20   }
21
22   private static void opprettSpekulanter(){
23
24      int i;
25      for (i=0; i<S; i++)
26        new Balanse_3().start();
27   }
28
29   public void run()  {
30
31     while(true)
```

```
32        flyttPenger();
33    }
34
35    private static synchronized void flyttPenger() {
36
37      int til, fra, belop;
38
39      fra=Math.abs( slumptall.nextInt() ) %N;
40      til=Math.abs( slumptall.nextInt() ) %N;
41      belop=Math.abs( slumptall.nextInt() ) % ( (konto[fra]/10) );
42
43      konto[fra]-=belop;
44      konto[til]+=belop;
45    }
46
47    private static synchronized void opprettKonti() throws
          InterruptedException{
48
49      int i;
50
51      for (i=0;i<N;i++)
52        konto[i]=T/N;
53
54      skriv_saldo();
55    }
56
57    private static synchronized void skriv_saldo() throws
          InterruptedException{
58
59      int sum, i;
60
61      for (i=0, sum=0; i<N; i++){
62        System.out.printf("konto_%d:\t%d\n", i, konto[i]);
63        sum+=konto[i];
64      }
65
66      System.out.printf("----------\nTotalt:\t%d\n==========\n",sum);
67      System.out.flush();
68    }
69  }
```

2.4 Øvelsesoppgave

Oppgave 2.1

- Under finner du koden til TCPSkravler_3. Denne følger ikke samme protokollen som TCPSkravler_2. TCPSkravler_2 skrev en linje, for hver linje klienten skrev. TCPSkravler_3 skriver en eller flere linjer. Lag en klient til Skravler_3. Tips bruk tråder!

- TCPSkravler_3.java

Listing 2.10: eksempler/TCPSkravler_3.java

```
1 import java.io.IOException;
```

```
2  import java.io.PrintWriter;
3  import java.net.ServerSocket;
4  import java.net.Socket;
5  import java.util.Scanner;
6
7
8  public class TCPSkravler_3 extends Thread{
9    public  Socket forbindelse;
10
11   public static void main(String[] args) throws IOException {
12
13     ServerSocket tjenerSocket = new ServerSocket(8888);
14     tjenerSocket.setReuseAddress(true);
15
16     while (true)
17       new TCPSkravler_3(tjenerSocket.accept()).start();
18   }
19
20   TCPSkravler_3(Socket s){
21
22     forbindelse = s;
23   }
24
25   public void run(){
26
27     PrintWriter utskriver;
28     Scanner innleser;
29     int i;
30
31     try {
32       utskriver = new PrintWriter(forbindelse.getOutputStream());
33       innleser = new Scanner(forbindelse.getInputStream());
34
35       while(innleser.hasNextLine()) {
36         for (i=0;i<(int)(Math.random()*3);i++)
37           utskriver.println("Hei, hei");
38         utskriver.format("Du sa:\t\"%s\".\n", innleser.nextLine() );
39         utskriver.flush();
40         }
41
42     } catch (IOException e) {
43       e.printStackTrace(); }
44
45     finally {
46         try {
47           forbindelse.close();
48
49         } catch (IOException e) {
50           e.printStackTrace();
51         }
52
53     }
54   }
55 }
```

Kapittel 3

Ikke-blokkerende-IO, URL-er, filer, kommandolinjeargumenter

3.1 Løsningsforslag på forrige øvelsesoppgave

- Oppgaven gikk ut på å lage en klient som kommuniserte med TCPSkravler_3. Protokollen er slik at klient sender en linje med tekst, og tjeneren svarer med en eller fere linjer. Ved å kjøre lesing og skriving i hver sin tråd, blir de uavhengige av hverandre – de trenger ikke vente på tur.

- TraadSkravleKlient.java

Listing 3.1: losningsforslag/TraadSkravleKlient.java

```
1  public class TraadSkravleKlient{
2
3    public static void main(String[] args) throws
4      java.net.UnknownHostException,
5      java.io.IOException {
6
7      java.net.Socket s;
8
9      s = new java.net.Socket(java.net.InetAddress.getByName("matiksi.hive.no
           "),8888);
10     new Dytter(s).start();
11     new Lytter(s).start();
12   }
13 }
```

- Lytter.java

Listing 3.2: losningsforslag/Lytter.java

```
1  public class Lytter extends Thread{
2
3    java.net.Socket s;
4    java.util.Scanner inn;
5
6    Lytter(java.net.Socket t) {
7      s=t;
8    }
```

```
9
10    public void run() {
11
12      try {
13
14        inn = new java.util.Scanner(s.getInputStream());
15
16        while (inn.hasNextLine())
17          System.out.println(inn.nextLine());
18
19      } catch (java.io.IOException e) {
20        e.printStackTrace();
21      }
22    }
23  }
```

- Dytter.java

Listing 3.3: losningsforslag/Dytter.java

```
1  public class Dytter extends Thread{
2
3    java.io.PrintWriter ut;
4    java.util.Scanner   inn;
5    java.net.Socket     s;
6
7    Dytter(java.net.Socket t) {
8      s=t;
9    }
10
11   public void run() {
12
13     try {
14
15       ut  = new java.io.PrintWriter(s.getOutputStream());
16       inn = new java.util.Scanner(System.in);
17
18       while (inn.hasNextLine()){
19         ut.println(inn.nextLine());
20         ut.flush();
21       }
22
23     } catch (java.io.IOException e) {
24       e.printStackTrace();
25     }
26   }
27 }
```

3.2 Mer trådsynkronisering

- Eksemplet TraadBryter.java demonsterer bruk av wait og notify. I eksemplet starter tre tråder. Trykk tallene 1,2 eller 3 for å suspendere/vekketrådene

- TraadBryter.java

Listing 3.4: eksempler/TraadBryter.java

```
1  public class TraadBryter extends Thread {
2
3    private boolean kjor;
4    static final int antTr = 3;
5
6    public static void main(String[] args) {
7
8      TraadBryter[] traader;
9      java.util.Scanner inn;
10     int i, n;
11
12     traader = new TraadBryter[antTr];
13
14     for (i=0; i<antTr; i++){
15       traader[i] = new TraadBryter();
16       traader[i].setName(String.format("Tr. %d", i));
17       traader[i].start();
18     }
19
20     inn = new java.util.Scanner(System.in);
21
22       while (inn.hasNext()){
23
24         n = inn.nextInt();
25         if (traader[n].getKjor() == false)
26           traader[n].setKjor(true);
27
28         else
29           traader[n].setKjor(false);
30       }
31   }
32
33   public synchronized void run() {
34     try {
35       kjor = true;
36       while (true){
37           wait(1500);
38           if (kjor){
39
40             System.out.printf("%s\n",this.getName());
41
42           } else
43             wait();
44         }
45
46     } catch (InterruptedException e) {
47       e.printStackTrace();
48     }
49   }
50
51   public synchronized void setKjor(boolean k) {
52     this.kjor = k;
53     if (kjor)
54       notify();
55   }
56
57   public boolean getKjor() {
```

```
58      return this.kjor;
59    }
60  }
```

3.3 Ikke-blokkerende tjener

- Eksemplet NonBlockSkravler.java demonsterer bruk av select og channels, fra javas NIO (New In- put/Output API).
- I stedet for javas tradisjonelle strømmer, bruker NIO kanaler (channels), som er blokkorienterte. Overføringen går fortere enn den tradisjonelle måten.
- Hver kanal registrerer i en selector, hvilke hendelser den er interessert i (accept, connect, read og/eller write).
- Multipleksing brukes her som et alternativ til å starte en tråd for hver klient-tilkobling. Det er en raskere, men litt mer komplisert, løsning.
- NonBlockSkravler.java

Listing 3.5: eksempler/NonBlockSkravler.java

```
1  import java.nio.channels.SelectionKey;
2
3  public class NonBlockSkravler {
4
5    private static java.nio.channels.Selector sel;
6    private static java.nio.channels.ServerSocketChannel sSC;
7
8    public static void main (String[] args) throws java.io.IOException {
9
10     java.net.InetSocketAddress adr;
11     java.util.Set hendelsesNokler;
12     SelectionKey nokkel;
13
14     int operasjoner;
15     java.util.Iterator it;
16
17     // Åpner en server-socket-kanal
18     sSC = java.nio.channels.ServerSocketChannel.open();
19
20     // Setter server-socket-kanalen til 'ikke-blokkerende'
21     sSC.configureBlocking(false);
22
23     // Henter en refereanse den server-socket som hører til
24     // server-socket-kanalen.
25     java.net.ServerSocket sS = sSC.socket();
26
27     // Knytter server-socketen til port 8888.
28     adr = new java.net.InetSocketAddress(8888);
29     sS.bind(adr);
30
31     // Åpner en selector
32     sel = java.nio.channels.Selector.open();
```

```
33
34      // Registrerer operasjonen accept for server-socket-kanalen
35      // i selectoren.
36      sSC.register(sel, SelectionKey.OP_ACCEPT);
37
38
39      while (true){
40
41        // select() Returnerer når minst
42        // en registert kanal valgt og klar for I/O.
43
44        sel.select();
45
46        // Henter et sett (java.util.Set) med valgte
47        // nøkler (java.nio.channels.SelctionKey)
48        hendelsesNokler = sel.selectedKeys();
49
50        // Henter referanse til settets iterator.
51
52        it = hendelsesNokler.iterator();
53
54        while(it.hasNext()) {
55
56          // Henter referanse til neste nøkkel.
57
58          nokkel = (SelectionKey) it.next();
59
60          // Henter et heltall med flagg som indikerer hvilke operasjoner
61          // som er registert som klare i nøkkelen
62
63          operasjoner = nokkel.readyOps();
64
65          if ((operasjoner & SelectionKey.OP_ACCEPT) == SelectionKey.
              OP_ACCEPT) {
66            tilkoble(nokkel);
67            continue;
68
69          }
70          if ((operasjoner & SelectionKey.OP_READ) == SelectionKey.OP_READ) {
71            lese(nokkel);
72            continue;
73          }
74        }
75      }
76    }
77
78    private static void tilkoble(SelectionKey nokkel) throws java.io.
        IOException {
79
80      java.nio.channels.SocketChannel sC;
81
82      /*
83       * Vi gjøre en accept() som returnerer med en gang. Den er ikke
             blokkerende.
84       * Siden vi er i metoden tilkoble(), vet vi at det er en forespørsel
             fra
85       * en klient som venter.
86       *
```

```
87        * En referense til en socket-kanal mot klienten returneres. Socket-
              kanalen
88        * settes til ikke-blokkerende. Lese-operasjoner for denne registreres
              i selectoren.
89        *
90        * Til slutt fjernes nøkkelen, som vi har fått referanse til v.h.a.
              argumentet 'nokkel',
91        * fra nøkkel-settet.
92        */
93
94       sC = sSC.accept();
95       sC.configureBlocking(false);
96       sC.register(sel, SelectionKey.OP_READ);
97       sel.selectedKeys().remove(nokkel);
98
99     }
100
101    private static void lese(SelectionKey nokkel) throws java.io.IOException
           {
102
103
104
105      /*
106       *  Ved hjelp av nøkkel-referansen i argumentet 'nokkel' kan vi hente
              en
107       *  referanse til socket-kanalen mot klienten.
108       */
109
110      java.nio.channels.SocketChannel sC = (java.nio.channels.SocketChannel)
             nokkel.channel();
111
112      /*
113       * Vi leser fra socket-kanalen med metoden read(). Denne vil returnere
              med en gang,
114       * siden kanalen er satt til ikke-blokkerende. Den skal inneholde data
              fra klienten
115       * siden vi er i metoden lese().
116       *
117       * Metoden read() skriver det den leser i et java.nio.ByteBuffer, så vi
               må klargjøre dette før
118       * vi kaller på read().
119       */
120
121      java.nio.ByteBuffer buffer = java.nio.ByteBuffer.allocate(2048);
122      buffer.clear();
123
124      if (sC.read(buffer) == -1)
125        sC.socket().close();
126
127      buffer.flip(); // limit=posisjon; posisjon=0;
128
129      /*
130       * En ikke-blokkerende SocketChannel vil skrive til
131       * utgående socket-buffer er fullt
132       * og så returnere.
133       *
134       * write() må derfor av og til kalles flere ganger.
135       */
```

```
136
137       while (buffer.remaining()>0)
138         sC.write(buffer);
139
140       sel.selectedKeys().remove(nokkel);
141
142     }
143   }
```

3.4 URL

- Eksemplet WebLaster.java demonsterer bruk av URL. For å lagre websiden på en fil kan følgende kommando kjøres fra kommandolinja: java WebLaster > filnavn.html.

- WebLaster.java

Listing 3.6: eksempler/WebLaster.java

```
1  public class WebLaster {
2
3    public static void main(String[] args) throws
4
5    java.net.MalformedURLException,
6    java.io.IOException{
7
8      java.io.InputStream innstrom;
9      java.util.Scanner skanner;
10     java.net.URL url;
11
12     url       = new java.net.URL("http://www.hive.no");
13     innstrom  = url.openStream();
14     skanner   = new java.util.Scanner(innstrom);
15
16     while (skanner.hasNext())
17       System.out.println(skanner.nextLine());
18
19   }
20
21 }
```

3.5 Lesing av fil

- Eksemplet Brukere.java demonsterer lesing av av kolon-separtert fil.

- Brukere.java

Listing 3.7: eksempler/Brukere.java

```
1  public class Brukere {
2
3    public static void main(String[] args) throws java.io.IOException {
4
5      java.util.Scanner innleser;
```

```
6       String [] post;
7       java.io.File filnavn;
8
9       filnavn =  new java.io.File("/etc/passwd");
10      innleser = new java.util.Scanner(filnavn);
11
12      while (innleser.hasNextLine()) {
13        post = innleser.nextLine().split(":");
14        if (post.length > 4) {
15          System.out.printf(
16               "Navn:\t%s\nLogin:\t%s\n\n",
17               post[4],
18               post[0] );
19        }
20      }
21      innleser.close();
22    }
23  }
```

3.6 Kommandolinje-argumenter

- Eksemplet Bruker.java demonsterer bruk av kommandolinje-argumenter.

- Bruker.java

Listing 3.8: eksempler/Bruker.java

```
1   public class Bruker {
2     public static void main(String[] args) throws java.io.IOException {
3
4       java.util.Scanner innleser;
5       String [] post;
6       java.io.File filnavn;
7
8       if (args.length < 1){
9         System.out.println("Du maa oppgi et brukernavn som argument.");
10        System.exit(1);
11      }
12
13      filnavn = new java.io.File("/etc/passwd");
14      innleser = new java.util.Scanner(filnavn);
15
16      while (innleser.hasNextLine()) {
17        post = innleser.nextLine().split(":");
18        if ( post.length > 4 && args[0].equals(post[0]) ) {
19          System.out.printf(
20               "Navn:\t%s\nLogin:\t%s\n\n",
21               post[4],
22               post[0] );
23        }
24      }
25      innleser.close();
26    }
27  }
```

3.7 Øvelsesoppgaver

Oppgave 3.1

- Skriv om WebLaster.java slik at websiden som lastes lagres i en fil. Både URL'n og filnavnet skal angis som kommandolinjeargumenter.

Oppgave 3.2

- Skriv om TCPSkravler slik at den lagrer alt klientene skriver til en logg-fil. Sørg for at hver linje kun inneholder tekst fra en klient. Hver linje bør også inneholde IP-adresse og portnummer. Lag også et testprogram for å teste at ikke teksten fra flere klienter blander seg på samme linje.

Oppgave 3.3

- Skriv om TCPSkravleTjeneren, slik at den skriver alt til alle klientene som er tilkoblet, slik at alle brukerne ser hva de andre skriver. Sørg for at hver linje kun inneholder tekst fra en klient.

Kapittel 4

RandomAccess-filtilgang, serialisering, vector

4.1 Løsningsforslag på forrige øvelsesoppgave

WebLaster

- I oppgaven skulle URL og filnavn komme som argumenter til main(). URL'n skulle lastes og lagres i filen.
- WebLaster_2.java

Listing 4.1: losningsforslag/WebLaster_2.java

```
1  public class WebLaster_2 {
2
3    public static void main(String[] args)
4    throws java.net.MalformedURLException, java.io.IOException {
5
6      java.io.PrintWriter ut;
7      java.util.Scanner inn;
8
9      if (args.length < 2) {
10       System.err.printf("Du må oppgi to argumenter.\n");
11       System.exit(1);
12     }
13
14     inn = new java.util.Scanner((new java.net.URL(args[0])).openStream());
15     ut  = new java.io.PrintWriter(new java.io.File(args[1]));
16
17     while (inn.hasNext())
18       ut.println(inn.nextLine());
19
20     ut.close();
21   }
22 }
```

TCPSkravler_4

- Referansene til sockets som er forbundet med klienter, lagres i en dynamisk liste av objekter (Vector). Denne listen ligger i et eget objekt med metoder for å legge

til ny socket, og å sende en tekst til alle sockets (+logg). Begge disse metodene er synchronized, slik at trådene kun kan benytte dem en om gangen.

- TCPSkravler_4.java

Listing 4.2: losningsforslag/TCPSkravler_4.java

```
1  public class TCPSkravler_4 extends Thread{
2
3    static Kringkaster kk;
4    int socketIndex;
5    java.net.Socket k;
6
7    public static void main(String[] args) throws java.io.IOException {
8
9      java.net.ServerSocket tjenerSocket = new java.net.ServerSocket(8888);
10     tjenerSocket.setReuseAddress(true);
11
12     kk  = new Kringkaster();
13
14     while (true) {
15       TCPSkravler_4 skr = new TCPSkravler_4();
16       skr.k = tjenerSocket.accept();
17       kk.leggTilKlient(skr.k);
18       skr.start();
19     }
20   }
21
22   public void run(){
23
24     java.io.PrintWriter ut;
25     java.util.Scanner inn;
26     String kAdr;
27
28     kAdr = String.format(
29         "%s.%s",
30         k.getInetAddress().toString(),
31         k.getPort()
32     );
33
34     try {
35
36       inn = new java.util.Scanner(k.getInputStream());
37       ut = new java.io.PrintWriter(k.getOutputStream());
38
39       ut.println("Hvem der?");
40       ut.flush();
41       if (inn.hasNext())
42         kAdr=inn.next()+"@"+kAdr;
43       ut.printf("Velkommen %s.\nSamtalen blir logget!\n",kAdr);
44       ut.flush();
45       kk.send(kAdr+" har ankommet.", "Skravler");
46       while(inn.hasNext())
47         kk.send(inn.nextLine() , kAdr);
48
49     }
50     catch (java.io.IOException e){
51       e.printStackTrace();
```

```
52      }
53    }
54  }
```

- Kringkaster.java

Listing 4.3: losningsforslag/Kringkaster.java

```
1  public class Kringkaster {
2
3    java.util.Vector<java.net.Socket> alleKlienter;
4    java.io.PrintWriter logg;
5
6    Kringkaster() throws java.io.FileNotFoundException{
7
8      alleKlienter = new java.util.Vector<java.net.Socket>();
9      logg = new java.io.PrintWriter(new java.io.File("logg"));
10   }
11
12   synchronized void send(String linje, String kAdr)
13   throws java.io.IOException {
14
15     java.io.PrintWriter ut;
16     String l = String.format("%s:\t%s\n", kAdr, linje);
17
18     for (java.net.Socket s:alleKlienter){
19
20       if (s.isConnected()){
21         ut=new java.io.PrintWriter(s.getOutputStream());
22
23         ut.printf(l);
24         ut.flush();
25       }
26       else {
27         s.close();
28         alleKlienter.remove(s);
29       }
30     }
31
32     logg.printf(l);
33     logg.flush();
34   }
35
36   synchronized int leggTilKlient(java.net.Socket s) {
37     alleKlienter.add(s);
38     return alleKlienter.size()-1;
39   }
40 }
```

SkravleTester

- Dette programmet kjører i gang flere tråder som kobler seg på skravletjeneren. Trådene skriver i full fart, for å teste om utskrift fra flere klienter blander seg sammen på samme line.

- SkravleTester.java

Listing 4.4: losningsforslag/SkravleTester.java

```
1  public class SkravleTester{
2
3    public static void main(String[] args) throws
4      java.net.UnknownHostException,
5      java.io.IOException {
6
7      java.net.Socket s;
8      int i;
9      for (i=0; i<10; i++){
10       s = new java.net.Socket(java.net.InetAddress.getByName("localhost")
             ,8888);
11       new AutoDytter(s).start();
12     }
13   }
14 }
```

- AutoDytter.java

Listing 4.5: losningsforslag/AutoDytter.java

```
1  public class AutoDytter extends Thread{
2
3    java.net.Socket s;
4
5    AutoDytter(java.net.Socket t) {
6      s=t;
7    }
8
9    public void run() {
10
11   try {
12       java.io.PrintWriter ut;
13       int i;
14       ut  = new java.io.PrintWriter(s.getOutputStream());
15
16       for (i=0; i<1000; i++){
17         ut.printf("%s - %d\n", this.getName(),i);
18         ut.flush();
19       }
20
21     } catch (java.io.IOException e) {
22       e.printStackTrace();
23     }
24   }
25 }
```

4.2 RandomAccessFile

- Eksemplet viser hvordan vi kan flytte en posisjonspeker, for lesing/og skriving, til en vilkårlig posisjon i fila. PersonLagrer_1 skriver poster med fast lengde til en fil, mens PersonHenter_1 henter frem en bestemt post fra fila.

- Person.java

Listing 4.6: eksempler/Person.java

```
1  class Person implements java.io.Serializable {
2
3    String navn;
4    int id;
5
6    public Person(int d, String s){
7      navn = s;
8      id   = d;
9    }
10
11   public String toString() {
12     return String.format("%d:\t%s\n", id, navn);
13   }
14 }
```

- PersonLagrer_1.java

Listing 4.7: eksempler/PersonLagrer_1.java

```
1  public class PersonLagrer_1 {
2
3    public static void main(String[] args)
4      throws java.io.IOException {
5
6      final int fastBredde = 8;
7
8      java.io.RandomAccessFile fil = new java.io.RandomAccessFile("person.dat
           ","rw");
9      java.util.Scanner inn= new java.util.Scanner(System.in);
10     int lengde,j;
11     String linje;
12
13     int i=0;
14
15     while(inn.hasNext()){
16
17       fil.writeInt(i++);
18       linje = inn.nextLine();
19       lengde = linje.length();
20
21       if ( lengde <= fastBredde ){
22         fil.writeChars(linje);
23         for (j=lengde; j<fastBredde; j++ )
24           fil.writeChar('␣');
25       } else
26         fil.writeChars(linje.substring(0, fastBredde));
27     }
28   }
29 }
```

- PersonHenter_1.java

Listing 4.8: eksempler/PersonHenter_1.java

```
1  public class PersonHenter_1 {
```

```
2
3   public static void main(String[] args) throws java.io.IOException {
4
5     final int fastBredde = 8;
6     final int postStr = 4 + fastBredde*2;
7
8     StringBuffer buf;
9     long postNr;
10    int id, i;
11
12    java.io.RandomAccessFile fil = new java.io.RandomAccessFile("person.dat
          ","r");
13    java.util.Scanner inn = new java.util.Scanner(System.in);
14    long antPost = fil.length()/postStr;
15
16    while (inn.hasNext()){
17
18      buf  = new StringBuffer();
19      postNr = inn.nextInt();
20
21      if (postNr >= antPost)
22        System.out.printf("Post␣%d␣finnes␣ikke.\n", postNr);
23
24      else {
25
26        fil.seek(postNr*postStr);
27        id = fil.readInt();
28
29        for (i=0; i<fastBredde; i++)
30          buf.append(fil.readChar());
31
32        System.out.printf("%d:␣%s\n", id, buf);
33      }
34    }
35  }
36 }
```

4.3 Serializable

- Ved å instansiere en klasse som *implements Serializable*, opprettes objekter som kan skrives i sin helhet direkte til en fil.

- PersonLagrer_2.java

Listing 4.9: eksempler/PersonLagrer_2.java

```
1 public class PersonLagrer_2 {
2
3   public static void main(String[] args) throws java.io.IOException {
4
5     java.io.ObjectOutputStream ut = new java.io.ObjectOutputStream(
6         new java.io.FileOutputStream("person_2.dat"));
7
8     java.util.Scanner inn = new java.util.Scanner(System.in);
9
10    int i=0;
```

```
11
12      while(inn.hasNext())
13        ut.writeObject(new Person(i++, inn.next()));
14
15      ut.close();
16    }
17  }
```

- PersonHenter_2.java

Listing 4.10: eksempler/PersonHenter_2.java

```
1  public class PersonHenter_2 {
2
3    public static void main(String[] args)
4    throws java.io.IOException, ClassNotFoundException {
5
6      java.io.ObjectInputStream inn = new java.io.ObjectInputStream(
7          new java.io.FileInputStream("person_2.dat"));
8
9      while (true){
10       try {
11         System.out.println( inn.readObject() );
12       }
13       catch (java.io.EOFException e){
14         inn.close();
15         break;
16       }
17     }
18   }
19 }
```

4.4 Vector

- Ved å lagre objekter som implementerer Serializable i en liste av klassen Vector, kan hele listen lagres (og hentes) i en operasjon.

- PersonLagrer_3.java

Listing 4.11: eksempler/PersonLagrer_3.java

```
1  public class PersonLagrer_3 {
2
3    public static void main(String[] args) throws java.io.IOException {
4
5      java.io.ObjectOutputStream ut = new java.io.ObjectOutputStream(
6          new java.io.FileOutputStream("person_3.dat"));
7
8      java.util.Scanner          inn = new java.util.Scanner(System.in);
9      java.util.Vector <Person> pv = new java.util.Vector<Person>();
10
11     int i=0;
12
13     while(inn.hasNext())
14       pv.add( new Person(i++, inn.next() ) );
```

```
15
16      ut.writeObject(pv);
17      ut.close();
18    }
19  }
```

- PersonHenter_3.java

Listing 4.12: eksempler/PersonHenter_3.java

```
1  public class PersonHenter_3 {
2
3    public static void main(String[] args)
4    throws java.io.IOException, ClassNotFoundException {
5
6      java.io.ObjectInputStream inn;
7      java.util.Vector <Person> pv;
8
9      inn = new java.io.ObjectInputStream(
10         new java.io.FileInputStream("person_3.dat"));
11
12     pv = (java.util.Vector<Person>) inn.readObject();
13
14     for (Person p:pv)
15         System.out.println(p);
16
17     inn.close();
18   }
19 }
```

4.5 Øvelsesoppgaver

Oppgave 4.1

- Lag en metode for å endre navnet på en person i fila person.dat. Metoden skal kun endre navnet (ikke skrive hele fila på nytt).

Oppgave 4.2

- Lag en PersonTjener og en PersonKlient, som kommuniserer v.h.a. TCP. Klienten oppgir en Person.id og PersonTjener returnerer et Person-objekt med matchende id.

Kapittel 5

HTML,CGI

5.1 Løsningsforslag til oppgaver fra forrige gang

4.1 - NavneEndrer

- Lag en metode for å endre navnet på en person i fila person.dat. Metoden skal kun endre navnet (ikke skrive hele fila på nytt).

- NavneEndrer.java

Listing 5.1: losningsforslag/NavneEndrer.java

```
1  public class NavneEndrer {
2
3    public static void main(String[] args) throws java.io.IOException {
4
5      final int fastBredde = 8;
6
7      // Setter post-størrelse
8      final int postStr = 4 + fastBredde*2;
9
10     if (args.length<2)
11       System.exit(1); // Avslutt med feilkode
12
13     // Tolk første argument som et heltall
14     int           id = Integer.parseInt(args[0]);
15
16     // Åpne fil med 'random access'
17     java.io.RandomAccessFile f = new java.io.RandomAccessFile("person.dat",
           "rw");
18
19     // Beregner antall poster
20     long          antPost = f.length()/postStr;
21
22     // Gjennomløper alle poster i fila
23     for (int i=0; i<antPost; i++){
24
25       f.seek(i*postStr);
26
27       if (id == f.readInt()) { // Har vi funnet riktig post?
28
29         if (args[1].length() <= fastBredde){
```

```
30            f.writeChars(args[1]);
31
32            // "Padding"
33            for (int j=args[1].length(); j<fastBredde; j++ )
34              f.writeChar('␣');
35          }
36          else
37            f.writeChars(args[1].substring(0, fastBredde));
38        }
39      }
40    }
41  }
```

4.2 - PersonKlient og PersjonTjener

- Lag en PersonTjener og en PersonKlient, som kommuniserer v.h.a. TCP. Klienten oppgir en Person.id og PersonTjener returnerer et Person-objekt med matchende id.

- PersonKlient.java

Listing 5.2: losningsforslag/PersonKlient.java

```
1  public class PersonKlient {
2
3      public static void main(String[] args)
4      throws
5      java.net.UnknownHostException, java.io.IOException,
6      ClassNotFoundException {
7
8        java.net.Socket           s = new java.net.Socket("localhost", 8889);
9        java.io.PrintWriter       sUt = new java.io.PrintWriter(s.
             getOutputStream());
10       java.util.Scanner       inn = new java.util.Scanner(System.in);
11       java.io.ObjectInputStream  sInn = new java.io.ObjectInputStream(s.
             getInputStream());
12
13       while (inn.hasNext()){
14
15         // Leser heltall og skriver til tjener
16         sUt.println(inn.nextInt());
17         sUt.flush();
18
19         inn.nextLine();       // Blir kvitt linjeskiftet
20
21         // Leser objekt, kaster det om til Person og skriver det ut.
22         System.out.println( (Person)sInn.readObject() );
23       }
24     }
25 }
```

- PersonTjener.java

Listing 5.3: losningsforslag/PersonTjener.java

```
1  public class PersonTjener {
2
```

```
3   public static void main(String[] args)
4   throws java.io.IOException, ClassNotFoundException {
5
6     java.net.ServerSocket ss = new java.net.ServerSocket(8889);
7     java.io.ObjectInputStream fInn;
8     java.io.ObjectOutputStream sUt;
9     java.util.Scanner sInn;
10    java.net.Socket s;
11    Person p;
12    int id;
13
14    while (true) {
15
16      // Blokkerer tråden. Returnerer med socket forbundet med klient.
17      s = ss.accept();
18
19      // ObjectOutputStream for skriving av objekter til klient
20      sUt  = new java.io.ObjectOutputStream(s.getOutputStream());
21
22      // Scanner for lesing av heltall fra klient
23      sInn = new java.util.Scanner(s.getInputStream());
24
25      while(sInn.hasNext()){
26
27        // ObjectInputStream for lesing av objekter fra fil
28        fInn = new java.io.ObjectInputStream(
29            new java.io.FileInputStream("person_2.dat"));
30
31        id = sInn.nextInt();  // Lese heltall fra klient
32        sInn.nextLine();     // Blir kvitt linjeskift
33
34        while(true) { // Søker basert på id (heltallet)
35
36          try {
37            p = (Person)fInn.readObject();
38            if (p.id == id) {
39              sUt.writeObject(p);
40              break;
41            }
42          }
43
44          catch (java.io.EOFException e){
45            sUt.writeObject(new Person(-1,"-"));
46            fInn.close();
47            break;
48          }
49        }
50      }
51    }
52  }
53 }
```

5.2 Grunnlenggende HTML

- Eksemplet viser oppbygningen av et html-dokument med hode og kropp.
- grunnleggende.html

Listing 5.4: eksempler/grunnleggende.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

<html>

  <head>

    <title> Tittel </title>

  </head>


  <body>

    <h1> Overskrift </h1>

    Vanlig tekst.
    Linjeskift må kodes med f.eks. <p> eller <br>.

    Vi kan referere til andre URL'er med <i>hyperlinker</i> som denne:

    <a href="http://www.hive.no">

    Hyperlink

    </a>

  </body>

</html>
```

5.3 Grunnleggende CGI

Hallo_1

- Demonstrasjon av hvordan deler av en URL brukes som input til CGI-program, via miljøvariablelen QUERY_STRING.

- hallo_1.html

Listing 5.5: eksempler/hallo_1.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

<html><head><title>Hallo_1</title></head>
  <body>

    Si hallo til:
    <ul>

      <li>
        <a href="hallo_1.cgi?Arne">Arne</a>
      <li>
        <a href="hallo_1.cgi?Bjarne">Bjarne</a>
      <li>
```

```
14          <a href="hallo_1.cgi?Cathrine">Cathrine</a>
15        <li>
16          <a href="hallo_1.cgi?Dolly">Dolly</a>
17
18      </ul>
19
20    </body>
21  </html>
```

- hallo_1.cgi

Listing 5.6: eksempler/hallo_1.cgi

```
1  #!/bin/sh
2
3  java -Dfile.encoding=UTF-8 Hallo_1
```

- Hallo_1.java

Listing 5.7: eksempler/Hallo_1.java

```
1  public class Hallo_1 {
2
3    public static void main(String[] args) {
4
5      // Skriver ut 'http-header' for 'plain-text'
6      System.out.printf("Content-type:text/plain;charset=utf-8\n\n");
7
8      // Skriver ut 'http-body'
9      System.out.printf(
10         "Hallo␣%s.\nHa␣en␣fin␣dag␣:)",
11         System.getenv("QUERY_STRING")
12     );
13
14   }
15 }
```

Hallo_2

- Demonstrasjon av hvordan et HTML-skjema brukes som input til CGI-program, via miljøvariablelen QUERY_STRING.

- hallo_2.html

Listing 5.8: eksempler/hallo_2.html

```
1  <!DOCTYPE HTML PUBLIC "-//W3C//DTD␣HTML␣4.01␣Transitional//EN">
2
3  <html><head><title>Hallo_2</title></head>
4    <body>
5      <form action=hallo_2.cgi>
6       <input type=text name=navn>
7       <input type=submit>
8      </form>
9    </body>
10 </html>
```

- hallo_2.cgi

Listing 5.9: eksempler/hallo_2.cgi

```
#!/bin/sh

java -Dfile.encoding=UTF-8 Hallo_2
```

- Hallo_2.java

Listing 5.10: eksempler/Hallo_2.java

```
public class Hallo_2 {

  public static void main(String[] args) {

    System.out.printf("Content-type:text/plain;charset=utf-8\n\n");

    System.out.printf(
        "Variabelen QUERY_STRING:\t%s",
        System.getenv("QUERY_STRING")
    );

  }
}
```

Hallo_3

- I dette eksemplet brukes også et HTML-skjema som input til et CGI-program. CGI-programmet dekoder QUERY_STRING. Den skriver dessuten html-kode, i motsetning til de foregående eksemplene som skrev ut ren tekst.

- hallo_3.html

Listing 5.11: eksempler/hallo_3.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html><head><title>Hallo_3</title></head>
  <body>
    <form action=hallo_3.cgi>
     <input type=text name=navn>
     <input type=submit>
    </form>
  </body>
</html>
```

- hallo_3.cgi

Listing 5.12: eksempler/hallo_3.cgi

```
#!/bin/sh

java -Dfile.encoding=UTF-8 Hallo_3
```

- Hallo_3.java

Listing 5.13: eksempler/Hallo_3.java

```
1  public class Hallo_3 {
2
3    public static void main(String[] args)
4    throws java.io.UnsupportedEncodingException {
5
6      // Skriver ut http-hode for html-tekst
7      System.out.println("Content-type:text/html;charset=utf-8\n\n");
8
9      // Skriver ut html-hodet
10     System.out.println(
11         "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.01 Transitional//EN\">"
               +
12         "<html><head><title>Hallo_3</title></head>"
13     );
14
15     // Skriver ut html-kroppen
16     System.out.printf(
17         "<body>Variabelen QUERY_STRING:\t%s</body>",
18         java.net.URLDecoder.decode(
19             System.getenv("QUERY_STRING"),
20             "UTF-8"
21         )
22     );
23
24     // Skriver ut avsluting av html-dokument
25     System.out.println("</html>");
26   }
27 }
```

Hallo_4

- I denne varianten skriver CGI-programmet ut skjemaet selv.

- hallo_4.cgi

Listing 5.14: eksempler/hallo_4.cgi

```
1  #!/bin/sh
2
3  java -Dfile.encoding=UTF-8 Hallo_4
```

- Hallo_4.java

Listing 5.15: eksempler/Hallo_4.java

```
1  import java.io.UnsupportedEncodingException;
2  import java.net.URLDecoder;
3
4  public class Hallo_4 {
5
6    public static void main(String[] args)
7    throws UnsupportedEncodingException {
```

```
8
9       // Skriver http-hodet
10      System.out.printf("Content-type:text/html\n\n");
11
12      // Skriver html-dokument med skjema
13      System.out.println(
14
15          "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.01 Transitional//EN\">"
                +
16          "<html><head><title>Hallo 4</title></head>" +
17          "<body>" +
18          "<form action=hallo_4.cgi>" +
19            "<input type=text name=fornavn>" +
20            "<br>" +
21              "<input type=text name=etternavn>" +
22              "<br>" +
23              "<input type=submit>" +
24            "</form>" +
25
26          "QUERY_STRING:" +
27          "<br>" +
28
29          URLDecoder.decode(
30               System.getenv("QUERY_STRING"),
31               "ISO-8859-1"
32          ) +
33
34          "</body></html>"
35      );
36
37    }
38  }
```

5.4 Øvelser

5.1 (Obligatorisk)

a)

- Lag en versjon av NavneEndrer (Oppgave 4.1) med et web-basert grensesnitt.

b)

- Lag en versjon av PersonKlient (Oppgave 4.2) med et web-basert grensesnitt.

5.2

- Løsningsforslaget på oppgave 4.2 over, har (minst) en svakhet. For hver gang programmet skal gjennomsøke fila, blir den åpnet og lukket. Skriv om løsningsforslaget, slik at tjeneren åpner fila kun en gang.

Kapittel 6

Oppsummering del 1

6.1 Løsningsforslag til oppgave fra forrige gang

- Vi hadde et løsningsforslag som besto av PersonTjener og PersonKlient, som kommuniserte v.h.a. TCP. Klienten oppga en Person.id og PersonTjener returnerte et Person-objekt med matchende id. Løsningsforslaget på oppgaven åpnet og lukket fila for hver gjennomsøkning. Oppgaven gikk ut på å skrive om løsningsforslaget, slik at fila kun ble åpnet en gang.

- Dette er løst i løsningsforslaget under, ved at alle objektene i fila lagres i minnet. De gjentatte søkene gjøres i minnet uten at fila blir involvert.

- PersonTjener_v2.java

Listing 6.1: losningsforslag/PersonTjener_v2.java

```
1  import java.io.EOFException;
2  import java.io.FileInputStream;
3  import java.io.IOException;
4  import java.io.ObjectInputStream;
5  import java.io.ObjectOutputStream;
6  import java.net.ServerSocket;
7  import java.net.Socket;
8  import java.util.Scanner;
9  import java.util.Vector;
10
11 public class PersonTjener {
12
13   public static void main(String[] args)
14   throws IOException, ClassNotFoundException {
15
16     ObjectInputStream fInn;
17     ObjectOutputStream sUt;
18     Vector<Person> pv;
19     ServerSocket ss;
20     Scanner sInn;
21     Socket s;
22
23     int pos, pvStr, id, i;
24
25     fInn = new ObjectInputStream(new FileInputStream("person_2.dat"));
26     ss = new ServerSocket(8889);
```

```
27      pv = new Vector<Person>();
28
29      while(true) {
30
31        try { pv.addElement((Person) fInn.readObject()); }
32
33        catch (EOFException e){
34          fInn.close();
35          break;
36        }
37      }
38
39      pvStr = pv.size();
40
41      while (true) {
42
43        s = ss.accept();
44
45        sUt = new ObjectOutputStream(s.getOutputStream());
46        sInn = new Scanner(s.getInputStream());
47
48        while(sInn.hasNext()){
49
50          id = sInn.nextInt();
51          sInn.nextLine();
52          pos = -1;
53
54          for (i=0; i < pvStr; i++)
55
56               if (pv.elementAt(i).id == id) {
57                 pos = i;
58                 break;
59               }
60
61          if (pos == -1)
62            sUt.writeObject(new Person(-1,"-"));
63          else
64            sUt.writeObject(pv.elementAt(pos));
65
66        }
67
68        s.close();
69      }
70    }
71  }
```

6.2 IP-adresse

IPv4-adresse: 32 bit

- dot quad: fire 8-bits nummer
- eks.: 128.39.105.10
- eksempler/MittNavnEr.java

Listing 6.2: eksempler/MittNavnEr.java

```
1  public class MittNavnEr {
2
3    public static void main(String[] args)
4
5    /*
6     * Unntaket java.net.UnknownHostException kastes ved mislykket
7     * navneoppslag.
8     *
9     * Med navneoppslag menes å fremskaffe ip-adressen til vert basert
10    * på vertens domenenavn.
11    *
12    * Både java.net.InetAddress.getLocalHost() og
13    * java.net.InetAddress.getByName(maskin) som benyttes i koden under
14    * kan kaste dette unntaket.
15    */
16
17   throws java.net.UnknownHostException {
18
19     java.util.Scanner inngang = new java.util.Scanner(System.in);
20
21     /*
22      * For å opprette et objekt av klassen java.net.InetAddress som
23      * refererer til "localhost" kan den statiske metoden
24      * java.net.InetAddress.getLocalHost() benyttes.
25      */
26
27     java.net.InetAddress adresse = java.net.InetAddress.getLocalHost();
28
29     System.out.println(
30         "Hallo, mitt navn er " +
31         adresse +
32         ". Skriv navnet på en annen maskin:"
33         );
34
35     String maskin = inngang.next();
36
37     /*
38      * For å gjøre et navneoppslag kan den statiske metoden
39      * java.net.InetAddress.getByName(String vert) benyttes.
40      *
41      * Dersom argumentet inneholder et domenenavn,
42      * vil metoden be vertsoperativsystemet gjøre et navneoppslag.
43      * (i DNS eller filen hosts).
44      *
45      *  Dersom 'vert' inneholder en tekststreng med ip-adresse,
46      *  gjøres kun en sjekk på om det er på riktig format.
47      */
48
49     adresse = java.net.InetAddress.getByName(maskin);
50
51     System.out.println(
52         "Adressen til " +
53         maskin +
54         " er " +
55         adresse +
56         "."
57         );
```

```
58     }
59 }
```

6.3 Sockets

TCP - klient/tjener i java

TCP-tjener

1. Opprette 'ServerSocket'

- ServerSocket serverSocket = new ServerSocket(PORTNUMMER);

2. Sette tjener i vente-tilstand

- Socket socket = serverSocket.accept();

3. Sette opp inn- og ut- strømmer

- socket.getInputStream();
- socket.getOutputStream();

4. Sende og motta data

- I henhold til applikasjons-protokoll

5. Stenge forbindelsen

- socket.close();

TCP-klient

1. Opprette forbindelse til tjener

- Socket socket = new Socket(INETADDR, PORT);

2. Sette opp inn- og ut- strømmer

- socket.getInputStream();
- socket.getOutputStream();

4. Sende og motta data

- I henhold til applikasjons-protokoll

5. Stenge forbindelsen

- socket.close();

Eksempel

- eksempler/TCPSkravler_1.java

Listing 6.3: eksempler/TCPSkravler_1.java

```
1  public class TCPSkravler_1 {
2
3    public static void main(String[] args)
4    throws java.io.IOException {
5
6      java.net.ServerSocket tjenerSocket = new java.net.ServerSocket(8888);
7      tjenerSocket.setReuseAddress(true);
8      java.io.PrintWriter utskriver;
9      java.io.InputStream innStrom;
10     java.net.Socket forbindelse;
11     java.util.Scanner innleser;
12
13     while (true) {
14
15       forbindelse = tjenerSocket.accept();
16
17       utskriver = new java.io.PrintWriter(forbindelse.getOutputStream());
18       innStrom = forbindelse.getInputStream();
19       innleser = new java.util.Scanner(innStrom);
20
21       while(true) {
22         utskriver.format(
23             "Interessant at du sier \"%s\".\n",
24             innleser.nextLine() );
25
26         utskriver.flush();
27       }
28     }
29   }
30 }
```

UDP - klient/tjener

UDP-tjener

1. Opprette 'DatagramSocket'-objekt

- DatagramSocket datagramSocket = new DatagramSocket(PORT);

2. Opprette byte-buffer

- byte[] buffer = new byte[BUFFERSTR];

3. Opprette 'DatagramPacket'-objekt

- DatagramPacket innPakke = new DatagramPacket(buffer, buffer.length);

4. Motta pakke

- datagramSocket.recieve(innPakke);

5. Finne avsenders adresse og portnr.

- InetAddress adr = innPakke.getAddress();
- int port = innPakke.getPort();

6. Hente ut pakkens data

- byte[] inndata = innPakke.getData();

7. Opprette datagram (for respons)

- DatagramPacket utPakke = new DatagramPacket(utData, utData.length(), adr, port);

8. Send datagram

- datagramSocket.send(utPakke);

9. Stenge 'DatagramSocket'

- datagramSocket.close();

UDP-klient

1. Opprette 'DatagramSocket'-objekt

- DatagramSocket datagramSocket = new DatagramSocket();

2. Opprette datagram

- DatagramPacket utPakke = new DatagramPacket(utData, utData.length(), adr, port);

3. Send datagram

- datagramSocket.send(utPakke);

4. Opprette byte-buffer

- byte[] buffer = new byte[BUFFERSTR];

5. Opprett (inn-)'DatagramPacket'-objekt

- DatagramPacket innPakke = new DatagramPacket(buffer, buffer.length);

6. Motta datagram

- datagramSocket.recieve(innPakke);

7. Hente ut data fra buffer

- byte[] inndata = innPakke.getData();

8. Stenge 'DatagramSocket'

- datagramSocket.close();

Eksempel

- eksempler/UDPSensor.java

Listing 6.4: eksempler/UDPSensor.java

```
1  public class UDPSensor {
2
3    public static void main(String[] args)
4    throws java.io.IOException {
5
6      java.net.DatagramSocket datagramSocket = new java.net.DatagramSocket
           (8888);
7      datagramSocket.setReuseAddress(true);
8
9      byte[] buffer = new byte[256];
10     java.net.DatagramPacket datagram = new java.net.DatagramPacket(buffer,
           buffer.length);
11
12     System.out.println(
13         "Tjeners adresse:\t"          + datagramSocket.getLocalAddress() +
14         "\nSensor aktivert på port\t" + datagramSocket.getLocalPort() );
15
16     datagramSocket.receive(datagram);
17
18     System.out.println(
19         "Datagrammet fra porten "    + datagram.getPort() +
20         " på maskinen "              + datagram.getAddress() +
21         " er detektert.");
22
23     datagramSocket.close();
24   }
25 }
```

6.4 Threads, synkronisering

Tråder

- En sekvens av kommandoer som utføres kalles en tråd
- En prosess (java vm) kan håndtere flere tråder på en gang
- Det er alltid minst en tråd om kjører, når et javaprogram kjøres
- Raskere å håndtere tråder enn prosesser
- Trådene deler et felles minneområde.
- Hver tråd har i tillegg eget minneormådet

en tråd er er et objekt som enten

- 'extends thread' eller
- implements runnable

Eksempler

- eksempler/TraadBryter.java

Listing 6.5: eksempler/TraadBryter.java

```
1  public class TraadBryter extends Thread {
2
3    private boolean kjor;
4    static final int antTr = 3;
5
6    public static void main(String[] args) {
7
8      TraadBryter[] traader;
9      java.util.Scanner inn;
10     int i, n;
11
12     traader = new TraadBryter[antTr];
13
14     for (i=0; i<antTr; i++){
15       traader[i] = new TraadBryter();
16       traader[i].setName(String.format("Tr. %d", i));
17       traader[i].start();
18     }
19
20     inn = new java.util.Scanner(System.in);
21
22       while (inn.hasNext()){
23
24         n = inn.nextInt();
25         if (traader[n].getKjor() == false)
26           traader[n].setKjor(true);
27
28         else
29           traader[n].setKjor(false);
30       }
31   }
32
33   public synchronized void run() {
34     try {
35       kjor = true;
36       while (true){
37           wait(1500);
38           if (kjor){
39
40             System.out.printf("%s\n",this.getName());
41
42           } else
43             wait();
44         }
45
```

```
46      } catch (InterruptedException e) {
47        e.printStackTrace();
48      }
49    }
50
51    public synchronized void setKjor(boolean k) {
52      this.kjor = k;
53      if (kjor)
54        notify();
55    }
56
57    public boolean getKjor() {
58      return this.kjor;
59    }
60  }
```

- eksempler/Balanse_3.java

Listing 6.6: eksempler/Balanse_3.java

```
1  public class Balanse_3 extends Thread{
2
3    static final int T = 1000; // Totalsummen i spillet
4    static final int S = 10;   // Antall spekulanter som flytter penger
5    static final int N = 5;    // Antall konti
6
7    static int konto[] = new int[N];
8
9    static java.util.Random slumptall = new java.util.Random();
10
11   public static void main(String[] args) throws InterruptedException  {
12
13     opprettKonti();
14     opprettSpekulanter();
15
16     while (true){
17       skriv_saldo();
18       sleep(1000);
19     }
20   }
21
22   private static void opprettSpekulanter(){
23
24      int i;
25      for (i=0; i<S; i++)
26        new Balanse_3().start();
27   }
28
29   public void run()  {
30
31     while(true)
32       flyttPenger();
33   }
34
35   private static synchronized void flyttPenger() {
36
37     int til, fra, belop;
38
```

```
39      fra=Math.abs( slumptall.nextInt() ) %N;
40      til=Math.abs( slumptall.nextInt() ) %N;
41      belop=Math.abs( slumptall.nextInt() ) % ( (konto[fra]/10) );
42
43      konto[fra]-=belop;
44      konto[til]+=belop;
45    }
46
47    private static synchronized void opprettKonti() throws
          InterruptedException{
48
49      int i;
50
51      for (i=0;i<N;i++)
52        konto[i]=T/N;
53
54      skriv_saldo();
55    }
56
57    private static synchronized void skriv_saldo() throws
          InterruptedException{
58
59      int sum, i;
60
61      for (i=0, sum=0; i<N; i++){
62        System.out.printf("konto %d:\t%d\n", i, konto[i]);
63        sum+=konto[i];
64      }
65
66      System.out.printf("----------\nTotalt:\t%d\n==========\n",sum);
67      System.out.flush();
68    }
69  }
```

6.5 Ikke-blokkerende-IO

- eksempler/NonBlockSkravler.java

Listing 6.7: eksempler/NonBlockSkravler.java

```
1  import java.nio.channels.SelectionKey;
2
3  public class NonBlockSkravler {
4
5    private static java.nio.channels.Selector sel;
6    private static java.nio.channels.ServerSocketChannel sSC;
7
8    public static void main (String[] args) throws java.io.IOException {
9
10     java.net.InetSocketAddress adr;
11     java.util.Set hendelsesNokler;
12     SelectionKey nokkel;
13
14     int operasjoner;
15     java.util.Iterator it;
16
```

```
17      // Åpner en server-socket-kanal
18      sSC = java.nio.channels.ServerSocketChannel.open();
19
20      // Setter server-socket-kanalen til 'ikke-blokkerende'
21      sSC.configureBlocking(false);
22
23      // Henter en refereanse den server-socket som hører til
24      // server-socket-kanalen.
25      java.net.ServerSocket sS = sSC.socket();
26
27      // Knytter server-socketen til port 8888.
28      adr = new java.net.InetSocketAddress(8888);
29      sS.bind(adr);
30
31      // Åpner en selector
32      sel = java.nio.channels.Selector.open();
33
34      // Registrerer operasjonen accept for server-socket-kanalen
35      // i selectoren.
36      sSC.register(sel, SelectionKey.OP_ACCEPT);
37
38
39      while (true){
40
41        // select() Returnerer når minst
42        // en registert kanal valgt og klar for I/O.
43
44        sel.select();
45
46        // Henter et sett (java.util.Set) med valgte
47        // nøkler (java.nio.channels.SelctionKey)
48        hendelsesNokler = sel.selectedKeys();
49
50        // Henter referanse til settets iterator.
51
52        it = hendelsesNokler.iterator();
53
54        while(it.hasNext()) {
55
56          // Henter referanse til neste nøkkel.
57
58          nokkel = (SelectionKey) it.next();
59
60          // Henter et heltall med flagg som indikerer hvilke operasjoner
61          // som er registert som klare i nøkkelen
62
63          operasjoner = nokkel.readyOps();
64
65          if ((operasjoner & SelectionKey.OP_ACCEPT) == SelectionKey.
                OP_ACCEPT) {
66            tilkoble(nokkel);
67            continue;
68
69          }
70          if ((operasjoner & SelectionKey.OP_READ) == SelectionKey.OP_READ) {
71            lese(nokkel);
72            continue;
73          }
```

```
74          }
75        }
76    }
77
78    private static void tilkoble(SelectionKey nokkel) throws java.io.
          IOException {
79
80      java.nio.channels.SocketChannel sC;
81
82      /*
83       * Vi gjøre en accept() som returnerer med en gang. Den er ikke
             blokkerende.
84       * Siden vi er i metoden tilkoble(), vet vi at det er en forespørsel
             fra
85       * en klient som venter.
86       *
87       * En referense til en socket-kanal mot klienten returneres. Socket-
             kanalen
88       * settes til ikke-blokkerende. Lese-operasjoner for denne registreres
             i selectoren.
89       *
90       * Til slutt fjernes nøkkelen, som vi har fått referanse til v.h.a.
             argumentet 'nokkel',
91       * fra nøkkel-settet.
92       */
93
94      sC = sSC.accept();
95      sC.configureBlocking(false);
96      sC.register(sel, SelectionKey.OP_READ);
97      sel.selectedKeys().remove(nokkel);
98
99    }
100
101   private static void lese(SelectionKey nokkel) throws java.io.IOException
          {
102
103
104
105     /*
106      *  Ved hjelp av nøkkel-referansen i argumentet 'nokkel' kan vi hente
             en
107      *  referanse til socket-kanalen mot klienten.
108      */
109
110     java.nio.channels.SocketChannel sC = (java.nio.channels.SocketChannel)
          nokkel.channel();
111
112     /*
113      * Vi leser fra socket-kanalen med metoden read(). Denne vil returnere
             med en gang,
114      * siden kanalen er satt til ikke-blokkerende. Den skal inneholde data
             fra klienten
115      * siden vi er i metoden lese().
116      *
117      * Metoden read() skriver det den leser i et java.nio.ByteBuffer, så vi
              må klargjøre dette før
118      * vi kaller på read().
119      */
```

```
120
121     java.nio.ByteBuffer buffer = java.nio.ByteBuffer.allocate(2048);
122     buffer.clear();
123
124     if (sC.read(buffer) == -1)
125       sC.socket().close();
126
127     buffer.flip(); // limit=posisjon; posisjon=0;
128
129     /*
130      * En ikke-blokkerende SocketChannel vil skrive til
131      * utgående socket-buffer er fullt
132      * og så returnere.
133      *
134      * write() må derfor av og til kalles flere ganger.
135      */
136
137     while (buffer.remaining()>0)
138       sC.write(buffer);
139
140     sel.selectedKeys().remove(nokkel);
141
142   }
143 }
```

6.6 filer, URL-er kommandolinje, argumenter

- losningsforslag/WebLaster_2.java

Listing 6.8: losningsforslag/WebLaster_2.java

```
1  public class WebLaster_2 {
2
3    public static void main(String[] args)
4    throws java.net.MalformedURLException, java.io.IOException {
5
6      java.io.PrintWriter ut;
7      java.util.Scanner inn;
8
9      if (args.length < 2) {
10       System.err.printf("Du må oppgi to argumenter.\n");
11       System.exit(1);
12     }
13
14     inn = new java.util.Scanner((new java.net.URL(args[0])).openStream());
15     ut  = new java.io.PrintWriter(new java.io.File(args[1]));
16
17     while (inn.hasNext())
18       ut.println(inn.nextLine());
19
20     ut.close();
21   }
22 }
```

6.7 RandomAccess-filtilgang, serialisering, vector

- eksempler/PersonLagrer_3.java

Listing 6.9: eksempler/PersonLagrer_3.java

```
1  public class PersonLagrer_3 {
2
3    public static void main(String[] args) throws java.io.IOException {
4
5      java.io.ObjectOutputStream ut = new java.io.ObjectOutputStream(
6          new java.io.FileOutputStream("person_3.dat"));
7
8      java.util.Scanner          inn = new java.util.Scanner(System.in);
9      java.util.Vector <Person> pv = new java.util.Vector<Person>();
10
11     int i=0;
12
13     while(inn.hasNext())
14       pv.add( new Person(i++, inn.next() ) );
15
16     ut.writeObject(pv);
17     ut.close();
18   }
19 }
```

- eksempler/PersonHenter_3.java

Listing 6.10: eksempler/PersonHenter_3.java

```
1  public class PersonHenter_3 {
2
3    public static void main(String[] args)
4    throws java.io.IOException, ClassNotFoundException {
5
6      java.io.ObjectInputStream inn;
7      java.util.Vector <Person> pv;
8
9      inn = new java.io.ObjectInputStream(
10         new java.io.FileInputStream("person_3.dat"));
11
12     pv = (java.util.Vector<Person>) inn.readObject();
13
14     for (Person p:pv)
15         System.out.println(p);
16
17     inn.close();
18   }
19 }
```

- eksempler/PersonHenter_1.java

Listing 6.11: eksempler/PersonHenter_1.java

```
1  public class PersonHenter_1 {
2
```

```
3   public static void main(String[] args) throws java.io.IOException {
4
5     final int fastBredde = 8;
6     final int postStr = 4 + fastBredde*2;
7
8     StringBuffer buf;
9     long postNr;
10    int id, i;
11
12    java.io.RandomAccessFile fil = new java.io.RandomAccessFile("person.dat
          ","r");
13    java.util.Scanner inn = new java.util.Scanner(System.in);
14    long antPost = fil.length()/postStr;
15
16    while (inn.hasNext()){
17
18      buf  = new StringBuffer();
19      postNr = inn.nextInt();
20
21      if (postNr >= antPost)
22        System.out.printf("Post␣%d␣finnes␣ikke.\n", postNr);
23
24      else {
25
26        fil.seek(postNr*postStr);
27        id = fil.readInt();
28
29        for (i=0; i<fastBredde; i++)
30          buf.append(fil.readChar());
31
32        System.out.printf("%d:␣%s\n", id, buf);
33      }
34    }
35  }
36 }
```

6.8 HTML, CGi

- eksempler/hallo_3.html

Listing 6.12: eksempler/hallo_3.html

```
1 <!DOCTYPE HTML PUBLIC "-//W3C//DTD␣HTML␣4.01␣Transitional//EN">
2 <html><head><title>Hallo_3</title></head>
3   <body>
4     <form action=hallo_3.cgi>
5      <input type=text name=navn>
6      <input type=submit>
7     </form>
8   </body>
9 </html>
```

- eksempler/hallo_3.cgi

Listing 6.13: eksempler/hallo_3.cgi

```
#!/bin/sh

java -Dfile.encoding=UTF-8 Hallo_3
```

- eksempler/Hallo_3.java

Listing 6.14: eksempler/Hallo_3.java

```
public class Hallo_3 {

  public static void main(String[] args)
  throws java.io.UnsupportedEncodingException {

    // Skriver ut http-hode for html-tekst
    System.out.println("Content-type:text/html;charset=utf-8\n\n");

    // Skriver ut html-hodet
    System.out.println(
        "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.01 Transitional//EN\">"
          +
        "<html><head><title>Hallo_3</title></head>"
    );

    // Skriver ut html-kroppen
    System.out.printf(
        "<body>Variabelen QUERY_STRING:\t%s</body>",
        java.net.URLDecoder.decode(
            System.getenv("QUERY_STRING"),
            "UTF-8"
        )
    );

    // Skriver ut avsluting av html-dokument
    System.out.println("</html>");
  }
}
```

6.9 Øvelser

Oppgave 6.1

- En tjener programmert i java, kan håndtere flere klienter samtidig ved hjelp av tråder eller selector (non-blocking io). Forklar hvordan de to teknikkene virker.

Oppgave 6.2

- Det finnes to mye brukte transport-protokoller for klient/tjener-applikasjoner på internett. Forklar hvordan et javaprogram som benytter den ene, skiller seg fra et som benytter seg av den andre.

Oppgave 6.3

Gi begrunnede eksempler på når det er naturlig å bruke

- java.io.RandomAccessFile
- java.io.File

Del II

Kapittel 7

RMI

7.1 Tradisjonelt to kommunikasjonsformer:

- overføring
- "fjernstyring"

7.2 Problem: Kan ikke lese minnet på fjern maskin. To måter å løse problemet på:

1.

- Som ved verdioverføring. Konvertere objektet til byte-sekvens og sende hele objektet. Objektet rekonstrueres på lokal maskin.
- Lokale endringer blir ikke automatisk reflektert i original-objektet.

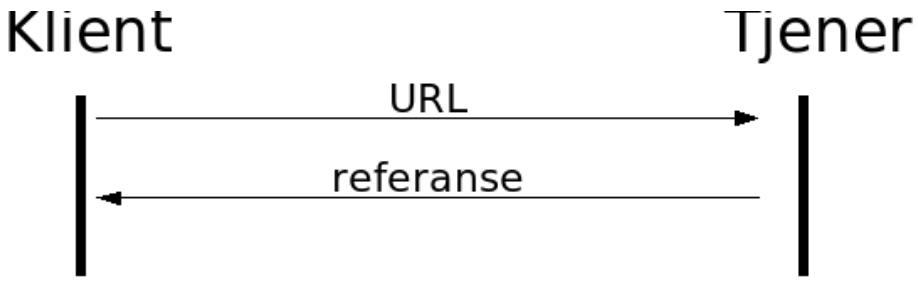
2.

- Som ved referanseoverføring. Remote reference
- Endringer reflekteres i begge ender.

7.3 Remote Method Invocation

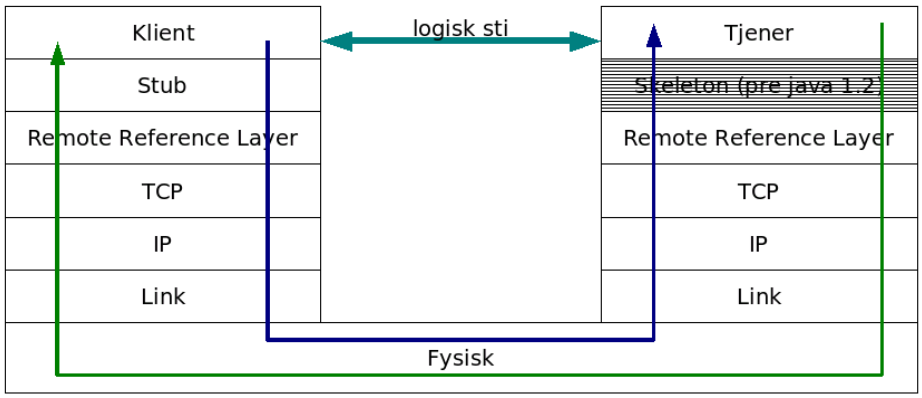
- Tjener registrerer et grensesnitt hos en navnetjener. Dette grensesnittet inneholder signaturene til de metodene som skal tilbys.
- Klienten gjør et navneoppslag i et register på en URL, og får en referanse tilbake.

figur



- Når klientene forespør tjenesten får den en referanse til et *remote object* i form av en *stub*. Et remote object implementerer et *remote interface*.
- Stubben videresender metodekall og parametre til en skeleton på det fjerne systemet.

figur



Parametre:

- Verdioverføring: Primitive typer.
- Referanseoverføring: Serialiserbare objekter (implements serializable) blir serialisert og kopiert (som verdioverføring).

7.4 Implementasjon av tjener og klient

Implementasjon av Tjener

Interface som extends remote

- Et markør-interface (marker interface) for å markere objekt som remote.
- Et remote object er definert som: En forekomst av en klasse som implementerer remote interface, eller et interface som extends Remote.
- Interface'et bestemmer hvilke metoder som er tilgjengelige utenfra.

- Hallogrensesnitt.java

Listing 7.1: eksempler/Hallogrensesnitt.java

```
1 public interface Hallogrensesnitt extends java.rmi.Remote{
2
3   public String hallo() throws java.rmi.RemoteException;
4
5 }
```

Implementasjon av interface

- Definere klasse som implementerer interface'et
- - må "stamme fra" java.rmi.server.RemoteObject. Vanligvis java.rmi.server.UnicastRemoteObject (enten direkte eller indirekte), som støtter punkt-til-punkt-forbindelser over TCP.
- - alternativt: Eksportere objektet som et fjernt objekt ved å sende det til en av de statiske UnicastRemoteObject.exportObject()-metodene.
- UnicastRemoteObject har metoder for å håndtere RMI. Bl.a. marshalling (argumenter og returverdier konverteres til byte-strøm) og unmarshalling (byte-strøm konverteres tilbake til argumenter og returverdier).
- Implementasjonen må inneholde en konstruktør som kan kaste Remote Exception. Fordi konstruktøren til UnicastRemoteObject kan det.
- Kun de metodene i klassen som implementerer interfacet, blir tilgjengelige utenfra.
- Den eneste forskjellen mellom en lokal og en fjern metode, er utenfor selve metodedefinisjonen. Forskjellen ligger i interfacet: Den fjerne metode er definert i interfacet; den lokale er det ikke.
- FjernHallo.java

Listing 7.2: eksempler/FjernHallo.java

```
1  public class FjernHallo
2  extends java.rmi.server.UnicastRemoteObject
3  implements Hallogrensesnitt{
4
5    public FjernHallo() throws java.rmi.RemoteException {
6
7    }
8
9    public String hallo() throws java.rmi.RemoteException {
10
11     return "Hallo";
12   }
13 }
```

Tjener-klassen

Tjeneren må gjøre to ting

- 1. Opprette objekt av klassen som implementerer interfacet
- 2. Knytte objekte til et navn (med Naming.bind() eller Naming.rebind()).
- Klienter kan spørre med navn eller etter en liste over tilgjengelige metoder.
- RmiTjener.java

Listing 7.3: eksempler/RmiTjener.java

```
1  public class RmiTjener {
2
3    public static void main (String[] args) throws
4    java.net.MalformedURLException,
5    java.rmi.RemoteException{
6
7      java.rmi.Naming.rebind(
8           "rmi://matiksi.hive.no/Hallo",
9           new FjernHallo()
10     );
11   }
12 }
```

- Det kan være flere instanser av samme klasse som er registrert med ulike navn. main() returnerer raskt, men tjeneren forstetter å kjøre...

Stubben

- Hvert fjernt objekt er representert lokalt som en stubb. Stubber inneholder informasjon fra remote interface. Java 1.5. kan noen ganger generere disse automatisk. Objekter som ikke er subklasser av UnicastRemoteObject, men eksportert med UnicastRemoteObject, må kompileres manuelt.
- Manuell kompilering kan gjøres med *rmic*. Ta med opsjonen *-v1.2*, hvis ikke blir *skeleton*-filen også laget. Input til rmic er klassefil som implementerer fjernt interfacet. Output er en stubb i form av en klassefil.

Oppstart av tjenesten

To tjenere må startes:

- 1. registeret
- 2. det fjerne objektet
- Registeret må startes først. Dette kan gjøres med kommandoen 'rmiregistry&'. Default port er 1099. For å starte registeret på port nummer $PORT, kan kommandoen 'rmiregistry $PORT& ' brukes. Hvis alternativ portnummer brukes, må dette inngå i URL'en som klienten bruker ved henvendelse til registertjenesten.

Klienten

- For at et objekt skal kunne kalle på en metode i et fjernt objekt, må det ha en "remote reference" til det fjerne objektet. Klienten skaffer seg dette, ved henvendelse til registeret på vertsmaskinen hvor det fjerne objektet finnes. Henvendelsen gjøres ved å kalle på registerets lookup()-metode.

- Objekt-referansen som returneres må kastes om til interfacet (klassen er ikke synlig for klienten). Objekter mister info om type når de lagres i strukturer som kan inneholde objekter av ulike klasser.

- URL'ene er bygget opp på samme måte som http-URL'er: rmi://vert.domene:port/navn

- Det fjerne objektet kan nå brukes som et lokalt. Den eneste forskjellen er at "RemoteException" må fanges ved bruk).

- Kompiler som vanlig - du trenger byte- /kilde- koden for interfacet (ikke klassen som implementerer den) tilgjengelig i klassestien.

- For å kjøre klienten må klassefilen for interfacet finnes i klassestien. (I "Pre java 1.5" trengs også stubb-klassefilen)

- RmiKlient.java

Listing 7.4: eksempler/RmiKlient.java

```
1  public class RmiKlient {
2
3    public static void main(String[] args) throws
4    java.net.MalformedURLException,
5    java.rmi.NotBoundException,
6    java.rmi.RemoteException
7    {
8      Hallogrensesnitt h =
9        (Hallogrensesnitt)java.rmi.Naming.lookup(
10           "rmi://matiksi.hive.no/Hallo");
11     System.out.println(h.hallo());
12   }
13
14 }
```

7.5 Øvelsesoppgave 07

- I denne oppgaven skal du skrive om løsningen på øvelse 04, slik slik at metodene beskrevet under kalles opp fra et "remote objekt", ved hjelp av "remote method invocation".

a)

- Lag en fjern metode for å endre navnet på en person i fila person.dat på tjeneren. Metoden skal kun endre navnet (ikke skrive hele fila på nytt).

b)

- Input til metode på fjernt objekt er Person.id. Metoden returnerer et Person-objekt med matchende id.

c)

- Bruk det du har lært om HTML og CGI til å lage klientprogrammet web-basert.

Kapittel 8

RMI II

8.1 Løsningsforslag fra forrige øvelse

a)

- Lag en fjern metode for å endre navnet på en person i fila person.dat på tjeneren. Metoden skal kun endre navnet (ikke skrive hele fila på nytt).

b)

- Input til metode på fjernt objekt er Person.id. Metoden returnerer et Person-objekt med matchende id.

- Person.java

Listing 8.1: losningsforslag/Person.java

```
1  import java.io.Serializable;
2
3
4  class Person implements Serializable {
5
6    int id;
7    String navn;
8
9    public Person(int d, String s){
10     navn = s;
11     id = d;
12   }
13
14   public String toString() {
15     return String.format("%d:\t%s\n", id, navn);
16   }
17 }
```

- RmiPersonregister.java (interface til tjeneren)

Listing 8.2: losningsforslag/RmiPersonregister.java

```
1  public interface RmiPersonregister extends java.rmi.Remote{
2
```

```
3   public void endreNavn(int id, String navn) throws
4   java.rmi.RemoteException, java.io.FileNotFoundException, java.io.
        IOException;
5
6   public Person getPerson(int id)throws java.rmi.RemoteException;
7   }
```

- RmiPersonTjener.java

Listing 8.3: losningsforslag/RmiPersonTjener.java

```
1  public class RmiPersonTjener
2  extends java.rmi.server.UnicastRemoteObject
3  implements RmiPersonregister, java.io.Serializable{
4
5    static java.io.RandomAccessFile f;
6    static java.util.Vector<Person> pv;
7
8    public RmiPersonTjener() throws java.rmi.RemoteException {
9    }
10
11   public static void main(String[] args)
12   throws java.io.IOException, ClassNotFoundException {
13
14     java.io.ObjectInputStream fInn;
15
16     fInn = new java.io.ObjectInputStream(new java.io.FileInputStream("
           person_2.dat"));
17     pv = new java.util.Vector<Person>();
18
19     java.rmi.Naming.rebind("rmi://matiksi.hive.no/Personregister", new
           RmiPersonTjener());
20
21     while(true) {
22
23       try { pv.addElement((Person) fInn.readObject()); }
24
25       catch (java.io.EOFException e){
26         fInn.close();
27         break;
28       }
29     }
30   }
31
32
33   public Person getPerson(int id) throws java.rmi.RemoteException {
34
35     for (Person p:pv)
36       if (p.id == id)
37         return p;
38
39     return new Person(-1,"-");
40   }
41
42   public void endreNavn(int id, String navn) throws java.rmi.
        RemoteException {
43
44     for (Person p:pv)
```

```
45        if (p.id == id)
46          p.navn=navn;
47    }
48  }
```

- RmiPersonKlient.java

Listing 8.4: losningsforslag/RmiPersonKlient.java

```
1  public class RmiPersonKlient {
2
3    public static void main(String[] args)
4    throws
5    java.net.UnknownHostException, java.io.IOException,
6    ClassNotFoundException, java.rmi.NotBoundException {
7
8      RmiPersonregister pt;
9      java.util.Scanner inn;
10
11     pt = (RmiPersonregister)java.rmi.Naming.lookup("rmi://matiksi.hive.no/
           Personregister");
12     inn = new java.util.Scanner(System.in);
13
14     while (inn.hasNext()){
15
16       System.out.println( pt.getPerson(inn.nextInt()));
17       inn.nextLine();
18     }
19   }
20 }
```

- RmiNavneEndrer.java

Listing 8.5: losningsforslag/RmiNavneEndrer.java

```
1  public class RmiNavneEndrer {
2
3    public static void main(String[] args)
4    throws
5    java.net.UnknownHostException, java.io.IOException,
6    ClassNotFoundException, java.rmi.NotBoundException {
7
8      RmiPersonregister pt;
9
10     if (args.length < 2)
11       System.exit(1);
12
13     pt = (RmiPersonregister)java.rmi.Naming.lookup("rmi://matiksi.hive.no/
           Personregister");
14
15     pt.endreNavn(Integer.parseInt(args[0]), args[1]);
16   }
17 }
```

8.2 RMI-Security

- En applikasjon får et objekt - mangler klassefila -prøver å laste den ned fra fjernt sted og instansiere objektet i sin JVM.
- Et objekt sendt som et RMI-argument kan initere kodekjøring umiddelbart etter deserialisering.
- Nedlasting av klassefiler håndteres av et objekt av klassen SecureClassLoader, som må ha sikkerhets restriksjoner definert.

RMISecurityManager

- extends SecurityManager
- forekomst kontrollerer implementasjon av Security Policy
- uten et slikt objekt vil ikke klasser lastes hvis de ikke er på lokalt filsystem

Eksempel 1

- Hallogrensesnitt.java

Listing 8.6: eksempler/Hallogrensesnitt.java

```
1 public interface Hallogrensesnitt extends java.rmi.Remote{
2
3   public String hallo() throws java.rmi.RemoteException;
4
5 }
```

- FjernHallo.java

Listing 8.7: eksempler/FjernHallo.java

```
1  public class FjernHallo
2  extends java.rmi.server.UnicastRemoteObject
3  implements Hallogrensesnitt{
4
5    public FjernHallo() throws java.rmi.RemoteException {
6
7    }
8
9    public String hallo() throws java.rmi.RemoteException {
10
11     return "Hallo";
12   }
13 }
```

- RmiTjener2.java

Listing 8.8: eksempler/RmiTjener2.java

```
1  public class RmiTjener2 {
2
3    public static void main (String[] args) throws
4    java.net.MalformedURLException,
5    java.rmi.RemoteException{
6
7      // alternativ til aa starte navneregisteret fra kommandolinjen (
           rmiregistry &)
8      java.rmi.registry.LocateRegistry.createRegistry(1099);
9
10     java.rmi.Naming.rebind(
11         "Hallo",
12         new FjernHallo()
13     );
14   }
15 }
```

- Banen til sikkerhetspolicy-fila kan angis som kommandolinje-argument (*-Djava.security.policy=kli*

- RmiKlient2.java

Listing 8.9: eksempler/RmiKlient2.java

```
1  public class RmiKlient2 {
2
3    public static void main(String[] args) throws
4    java.net.MalformedURLException,
5    java.rmi.NotBoundException,
6    java.rmi.RemoteException
7    {
8      // For aa tillate kjoering av nedlastet kode
9      System.setSecurityManager(new java.rmi.RMISecurityManager());
10
11     Hallogrensesnitt h =
12       (Hallogrensesnitt)java.rmi.Naming.lookup(
13           "rmi://matiksi.hive.no/Hallo");
14
15     System.out.println(h.hallo());
16   }
17
18 }
```

- klient.policy

Listing 8.10: eksempler/klient.policy

```
1  grant {
2      permission java.net.SocketPermission "matiksi.hive.no", "connect,
           resolve";
3  };
```

Eksempel 2

- Banen til sikkerhetspolicy-fila kan angis som kommandolinje-argument (*-Djava.security.policy=tje*
- RmiTjener3.java

Listing 8.11: eksempler/RmiTjener3.java

```
1  public class RmiTjener3 {
2
3    public static void main (String[] args) throws
4    java.net.MalformedURLException,
5    java.rmi.RemoteException{
6
7      System.setSecurityManager(new java.rmi.RMISecurityManager());
8
9      // alternativ til aa starte navneregisteret fra kommandolinjen (
           rmiregistry &)
10     java.rmi.registry.LocateRegistry.createRegistry(1099);
11
12     java.rmi.Naming.rebind(
13         "Hallo",
14         new FjernHallo()
15     );
16   }
17 }
```

- tjener.policy

Listing 8.12: eksempler/tjener.policy

```
1  grant {
2    permission java.net.SocketPermission "matiksi.hive.no", "connect,resolve"
         ;
3    permission java.net.SocketPermission "*", "accept";
4  };
```

8.3 Obligatiorisk oppgave

- a) Gjør om dagens løsningsforslag slik at det er et fjernt objekt for hver Person, i stedet for et felles objekt, slik det er beskrevet i avsnitt 5.4.Method2 (s.149...) i pensumboka. Klienten skal virke fra en annen maskin enn den tjeneren kjører på.

Kapittel 9

Servlets

9.1 Om servlets

- En *servlet* er et program skrevet i java som kjører på en web-tjener. Dette står i motsetning til en *applet* som kjøres på klienten.

Servlets container

- The container is the intermediary between the Web server and the servlets in the container. The container loads, initializes, and executes the servlets.

Tomcat

- tomcat

`http://tomcat.apache.org`

- Vi bruker *tomcat* som er en 'åpen kildekode'-implementasjon av *Java Servlet* and *JavaServer Pages* -teknologiene.

- På matiksi kjører Tomcat på port 8180.

- Tomcat er ikke lenger den offiselle referanseimplementasjonen, slik det står i pensumboka. Den rollen er overtatt av *glassfish*.

- Kjøringen trigges av http-forespørsel fra web-klient.

- Produserer dokument (f.eks. et html-dokument) som returneres til klienten.

9.2 Eksempler

Eksempel 1: ServletHallo

- ServletHallo

`http://matiksi.hive.no:8180/~thomas/ServletHallo`

```
thomas@hve6043:~/public_html/WEB-INF$ ls -l
totalt 12
drwxr-xr-x 3 thomas thomas 4096 2009-04-19 14:32 classes
drwxr-xr-x 2 thomas thomas 4096 2009-03-30 12:44 lib
-rw-r--r-- 1 thomas thomas 1681 2009-03-29 17:12 web.xml
thomas@hve6043:~/public_html/WEB-INF$
```

- WEB-INF

`http://matiksi.hive.no/~thomas/WEB-INF/`

- ServletHallo.java

Listing 9.1: eksempler/ServletHallo.java

```
1  public class ServletHallo  extends javax.servlet.http.HttpServlet {
2
3    // gjoer en http-get
4      public void doGet(
5          javax.servlet.http.HttpServletRequest  request,
6          javax.servlet.http.HttpServletResponse respons )
7
8          throws java.io.IOException,
9                 javax.servlet.ServletException {
10
11       // http-hodet
12       respons.setContentType("text/Plain");
13
14       java.io.PrintWriter ut = respons.getWriter();
15
16       // http-kroppen
17       ut.println("Hallo");
18         ut.flush();
19     }
20 }
```

- eksemper/web.xml

Listing 9.2: eksempler/web.xml

```
1  <?xml version="1.0" encoding="ISO-8859-1"?>
2  <web-app xmlns="http://java.sun.com/xml/ns/j2ee"
3      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4      xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://java.sun.com
           /xml/ns/j2ee/web-app_2_4.xsd"
5      version="2.4">
6
7  <servlet>
8   <servlet-name>ServletHallo</servlet-name>
9   <servlet-class>ServletHallo</servlet-class>
10 </servlet>
11
12 <servlet-mapping>
13  <servlet-name>ServletHallo</servlet-name>
14  <url-pattern>/ServletHallo</url-pattern>
15 </servlet-mapping>
16
17 </web-app>
```

Eksempel 2: Input fra html-skjema

- eksempler/serveline.html

Listing 9.3: eksempler/serveline.html

```
<html>
<form action="http://matiksi.hive.no:8180/~thomas/Serveline">
Brukernavn: <input type=text name=Brukernavn> <p>
<input type=submit>
</form>
</html>
```

- Serveline.java

Listing 9.4: eksempler/Serveline.java

```
public class Serveline  extends javax.servlet.http.HttpServlet {

    public void doGet(
        javax.servlet.http.HttpServletRequest  request,
        javax.servlet.http.HttpServletResponse respons )

        throws java.io.IOException,
               javax.servlet.ServletException {
      java.io.PrintWriter ut = respons.getWriter();

        String brukernavn = request.getParameter("Brukernavn");


        respons.setContentType("text/Plain");
        ut.println("Hallo "+brukernavn+", jeg heter Serveline");

        ut.flush();
    }
}
```

Eksempel 3: Informasjonskapsler

- http://matiksi.hive.no:8180/ thomas/CookieTest

`http://matiksi.hive.no:8180/~thomas/CookieTest`

- For å se informasjonskapslene i firefox: Edit preferences -> privacy -> show cookies

- CookieTest.java

Listing 9.5: eksempler/CookieTest.java

```
public class CookieTest  extends javax.servlet.http.HttpServlet {

    public void doGet(
        javax.servlet.http.HttpServletRequest  request,
        javax.servlet.http.HttpServletResponse respons )

        throws java.io.IOException,
```

```
8                 javax.servlet.ServletException {
9
10        java.io.PrintWriter ut;
11        javax.servlet.http.Cookie nybaktKake;
12        javax.servlet.http.Cookie[] kake;
13        int i;
14
15        ut = respons.getWriter();
16        respons.setContentType("text/html");
17
18        kake = request.getCookies();
19        if (kake == null) {
20
21          nybaktKake = new javax.servlet.http.Cookie("IP", request.
                getRemoteAddr() );
22            nybaktKake.setMaxAge(60*60);
23            respons.addCookie(nybaktKake);
24            ut.println("Velkommen␣fremmede");
25
26        } else {
27          // skriver ut verdiene i informasjonskapslene
28          ut.println("Takk␣for␣sist!");
29          for (i=0; i<kake.length;i++)
30            ut.printf(
31                 "%s:␣%s<br>",
32                 kake[i].getName(),
33                 kake[i].getValue()
34            );
35        }
36
37          ut.flush();
38      }
39  }
```

Eksempel 4: Sesjoner

- SesjonsTest.java

Listing 9.6: eksempler/SesjonsTest.java

```
1  public class SesjonsTest  extends javax.servlet.http.HttpServlet {
2
3    public void doGet(
4        javax.servlet.http.HttpServletRequest  request,
5        javax.servlet.http.HttpServletResponse respons )
6
7    throws
8    java.io.IOException,
9    javax.servlet.ServletException {
10
11     java.io.PrintWriter ut;
12
13     javax.servlet.http.HttpSession session = request.getSession();
14
15     Integer ival = (Integer) session.getAttribute("sessiontest.counter");
16
17     if (ival==null)
```

```
18        ival = new Integer(1);
19      else
20        ival = new Integer(ival.intValue() + 1);
21      session.setAttribute("sessiontest.counter", ival);
22
23      ut = respons.getWriter();
24      respons.setContentType("text/html");
25
26      ut.println("<html><body>");
27      ut.println("Tellerverid: " + ival + "<p>");
28
29      if (session.isNew())
30        ut.println("Ny sesjon <p>SesjonsID:" + session.getId());
31      else
32        ut.println("Gammel sesjon.<p>SesjonsID:" + session.getId());
33
34
35          ut.println("<form><input type=submit></form></body></html>");
36      ut.flush();
37    }
38  }
```

- SesjonsTest

`http://matiksi.hive.no:8180/%7Ethomas/SesjonsTest`

- java.sun.com > Products > Servlet > 2.2 > Javadoc > Javax > Servlet > Http > HttpSession

`http://java.sun.com/products/servlet/2.2/javadoc/javax/servlet/http/HttpSession.html`

9.3 Øvelsesoppgaver

- Lag en servlet som ved første besøk fra en bruker gjør følgende: Henter brukernavn, passord, fornavn, etternavn fra et html-skjema. Opplysningene skal lagres i en informasjonskapsel (cookie) i brukerens webklient.

- Ved senere besøk, hentes informasjonen frem fra informasjonskapselen i brukerens nettleser, og vises frem for brukeren.

Kapittel 10

JavaServer Pages

10.1 Hva er JSP?

- HTML-dokumenter med innfelt javakode.
- Som en utvidelse av servlet-teknologien.
- fil-endelse: .jsp
- med JSP menes både selve web-sidene som lages og teknologien som brukes for å lage web-sidene

Eksempel:

- http://matiksi.hive.no:8180/ thomas/Hallo.jsp

`http://matiksi.hive.no:8180/~thomas/Hallo.jsp`

- eksempler/Hallo.jsp

Listing 10.1: eksempler/Hallo.jsp

```
1  <html>
2  <head>
3    <title>JSP-Hallo</title>
4  </head>
5  <body>
6
7    <%-- Dette er en kommentar --%>
8
9    <% out.println("Hallo"); %>
10
11 </body>
12 </html>
```

10.2 Hvorfor JSP ble introdusert?

- Man slipper alle *out.println()*-kommandoene for HTML-koden.

- JSP er en annen måte å skrive servlets uten å være "java-ekspert"
- I det opprinnelige JSP-API'et var det nødvendig med noe java-kode, men etterhvert er det utviklet et eget bibliotek med egne "tagger" - JavaServer Pages Standard Tag Library (JSTL)

JSTL

- iterasjoner
- betingelser
- XML
- internasjonalisering
- DB-access (m/SQL)
- Eget språk tilpasset web-designere/utviklere:*Expression Language* (EL)

Arbeidsdeling/spesialisering

- Ved å kombinere servlets og JSP, er det lettere å fordele oppgaver mellom f.eks. en web-designer/utvikler og en java-programmerer.
- I alle web-applikasjoner, er det programmer på tjenere som behandler forspørsler og genererer responser.
- 1. Behandling av forespørsel
- 2. Forretningslogikk
- 3. Presentasjon
- Selv en web-side-forfatter som jobber alene vil kunne dra nytte av en slik inndeling
- Spesialisert kunnskap kan bedre utnyttes ved en slik isolering av oppgavene

10.3 Når brukes JSP, og når brukes servlets?

- Programmer som ikke gir noe utskrift er gode kandiater til servlets

Servlets

- I servlets trengs kompentanse i javaprogrammering overalt
- F.eks ved nytt utseende, må javaprogrammet endres
- Vanskelig å dra nytte av webutviklings-verktøy
- Servlets bra for programmerere

10.4 kompilering/kjøring

- En web-tjener trenger en servlet-kontainer for å håndtere servlets, og en JSP-kontainer å håndtere JSP'er
- Vi bruker Tomcat som er både web-tjener, servlet-kontainer og JSP-kontainer.
- JSP-kontaineren avbryter forespørsel til JSPer og produserer servlet ved behov og sender forespørsel til servlet som er "mappet" til JSP'en.

JSP arver dermed alle fordelene med servlets

- Plattform- og leverandør-uavhengighet

Integrasjon

- JDBC
- RMI
- etc.

effektivitet

- forblir i minnet

skalerbarhet

- p.g.a. javas utbredelse i alt fra små til store systemer

robusthet/sikkerhet

- "strongly typed"
- automatisk minnehåndtering
- "JVM = sandkasse"
- JSP-filene ligger på samme sted som html-filene.
- JSP blir kompilert til servlets ved første klientforespørsel.
- Den kompilerte filen beholdes inntil tjeneren terminerer eller JSP-kildekoden endres.
- Dette fører til lang respostide ved første referanse.
- Prekompilering kan gjøres.
- Eksempel på prekompilering: http://matiksi.hive.no:8180/ thomas/Hallo.jsp?jsp_precompile

`http://matiksi.hive.no:8180/~thomas/Hallo.jsp?jsp_precompile`

- Det er ikke nødvendig med omstart av server, slik som med servlets.
- Ved forespørsel blir template-tekst og JSP-elementer slått sammen

10.5 JElementer

To typer elementer

template text

- Blir alltid sendt rett gjennom til nettleseren

Statisk innhold

- HTML
- plain-text
- XML
- etc.

JSP-elementer

- noen få JSP-elementer for dynamisk innhold

Kategorier

- 1. Direktiver
- 2. Deklarasjoner
- 3. Uttrykk / "expressions"
- 4. Scriptlets
- 5. Kommentarer
- 6. Handling (action)
- (7. "Expression Language"-uttrykk)

1. Direktiver

- Informasjon om selve siden (som forblir lik mellom forespørsler)

<%@ page ... %>

- Attributter tilhørende selve siden. F.eks:
- contentType (del av http-header)
- import (samme som i java)
- errorPage (se eksempel under)

<%@ include ... %>

- Setter inn en fil i løpet av oversettelsesfasen

<%@ taglib ... %>

- spesifiserer et "tag library"

2. Deklarasjoner

- <%! ... %>
- Eks: <%! int teller; %>
- Dette produserer instansvariabler til servlet-klassen, som kan brukes i etterfølgende JSP-tagger.

3. Uttrykk / "expressions"

- kodeuttrykk hvis resultat skal med i respons. eller ved forspørsel brukt om "action-attributt-verdi"
- <%= ... %>
- Eks.: <%= teller++ %>
- Legg merke til at det ikke er semikolon i uttrykket.

4. Scriptlets

- <% ... vanlig java kode ... %>
- Ikke bruk dette mye - ved behov for mye kode, skill ut i servlet eller java-bean (se neste forelsning).
- Variabel-deklarasjoner kan også gjøres i scriptlets.
- Variabler deklarert i scriptlets, kan også brukes i etterfølgende JSP-tags.

5. Kommentarer

- <%– ... –%>
- Kommentarer fjernes, slik at de ikke sendes til klientene.

6. Handling (action)

- Ufører funksjoner som utgjør en utvidelse av JSP-standarden, f.eks. v.h.a. Java Beans
- Åpnings taggen spesifiserer bibliotek- og handlings-navn, adskilt med kolon:
- Eksempel: <jsp:useBean>
- Vi skal se mer om dette i forelesningen om java-beans.

(7. "Expression Language"-uttrykk)

- ikke pensum

Et eksempel med kommentarer, direktiver og uttrykk:

- http://matiksi.hive.no:8180/ thomas/Klokka.jsp

`http://matiksi.hive.no:8180/~thomas/Klokka.jsp`

- eksempler/Klokka.jsp

Listing 10.2: eksempler/Klokka.jsp

```
1  <html>
2  <head>
3    <title>Direktiver</title>
4  </head>
5  <body>
6
7
8  <%-- Direktiv --%>
9  <%@ page import="java.util.Calendar" %>
10
11 <%-- Expressions --%>
12
13 Klokka er
14
15 <%= Calendar.getInstance().get(Calendar.MINUTE) %>
16
17 over
18
19 <%= Calendar.getInstance().get(Calendar.HOUR) %>
20
21 </body>
22 </html>
```

10.6 Implisitte JSP-objekter

- En web-tjener med JSP-støtte skal gi JSP'ene tilgang til noen ferdig deklarerte objekter som er instanser av klasser definert i servlet- og JSP- spesifkasjonen.

HttpServletRequest request

- http-forespørsel fra klienten

HttpServletResponse response

- http-responsen som skal sendes til klienten

HttpSession session

- Sesjonsobjekt forbundet med request- og response- objektene

ServletContext application

- har referanser til objekter tilgjengelig for mer enn en bruker f.eks. DB-forbindelse

JspWriter out

- For å skrive til "response output stream"
- ofte unødvendig

Throwable exception

- kun tilgjengelig i "error pages"

tre som er sjelden i bruk

- PageContext pageContext
- ServletConfig config
- Object page - "this"

10.7 Tilgang til DB via JDBC fra JSP

plassere nødvendig kode i JSP'en

Vi ser på et eksempel:

- http://matiksi.hive.no:8180/ thomas/DBConnect.jsp

`http://matiksi.hive.no:8180/~thomas/DBConnect.jsp`

- DBConnect.jsp

Listing 10.3: eksempler/DBConnect.jsp

```
1  <%@ page errorPage="Feilmelding.jsp" import="java.sql.*" %>
2
3  <HTML>
4  <HEAD>
5    <TITLE>DB-connect</TITLE>
6  </HEAD>
7  <BODY>
8
9  <%
10
11 Class.forName("com.mysql.jdbc.Driver");
12 Connection lnk = DriverManager.getConnection(
13   "jdbc:mysql://localhost/bokbase",
14   "db",
15   "ada"
16 );
17
18 ResultSet res;
```

```
19 res = lnk.createStatement().executeQuery("show tables");
20 %>
21
22 <PRE>
23
24 <% while(res.next())
25    out.println(res.getString(1));
26 %>
27
28 </PRE>
29 </BODY>
30 </HTML>
```

definere spesialtilpassede "tags"

- (ikke pensum)

bruke JavaBeans

- vi ser eksempel på dette i neste forelesning

10.8 To hovedmetoder for samarbeid mellom servlets og JSP

Applikasjonens miljø

- tilgjengelig via det implisitte objektet 'application'
- i servlet via ServletContext fra 'getServletContext()'

Brukernes sesjon

- tilgjengelig i servlet som 'Session'-objekt
- tilgjengelig i JSP via det implisitte objektet 'session'

Vi ser på eksempler:

- SesjonsTest2.java

Listing 10.4: eksempler/SesjonsTest2.java

```
1  import java.io.IOException;
2  import java.io.PrintWriter;
3  import javax.servlet.ServletException;
4  import javax.servlet.http.HttpServlet;
5  import javax.servlet.http.HttpServletRequest;
6  import javax.servlet.http.HttpServletResponse;
7  import javax.servlet.http.HttpSession;
8
9
10 public class SesjonsTest2  extends HttpServlet {
```

```
11
12      public void doGet( HttpServletRequest  request,
13                         HttpServletResponse respons )
14
15          throws IOException,
16                 ServletException {
17
18        PrintWriter ut;
19        Integer teller;
20
21        HttpSession session = request.getSession();
22        teller = (Integer) session.getAttribute("teller");
23         if (teller==null)
24             teller = new Integer(1);
25         else
26             teller = new Integer(teller.intValue() + 1);
27         session.setAttribute("teller", teller);
28
29        ut = respons.getWriter();
30        respons.setContentType("text/html");
31
32        ut.printf("Sesjonens identifikator (cookie med navnet JSESSIONID):<BR
              >%s<P>", session.getId() );
33        ut.printf("Sesjonens Integer med navn 'teller':<BR>%d<P>", teller);
34        ut.println("<A HREF=Sesjonsdata.jsp> Hyperlenke til JSP </A>");
35        ut.flush();
36      }
37
38
39  }
```

- Sesjonsdata.jsp

Listing 10.5: eksempler/Sesjonsdata.jsp

```
1  <HTML>
2  <HEAD>
3    <TITLE>Sesjonsdata i JSP</TITLE>
4  </HEAD>
5  <BODY>
6
7
8  Sesjonens identifikator (cookie med navnet JSESSIONID):
9  <BR>
10 <%= session.getId()%>
11
12 <P>
13
14 Sesjonens Integer med navn 'teller':
15 <BR>
16 <%= session.getAttribute("teller") %>
17
18 <P>
19
20 <A HREF=SesjonsTest2> Hyperlenke til servlet </A>
21
22 </BODY>
23 </HTML>
```

10.9 Error pages

- Bruke servlet til å generere en feilmeldings-side og redirigere kontrollen til denne

eller bruke JSP-spesifisert måte å håndtere feil

- <%@ page errorPage=" ....jsp " %>

i feilmeldings-JSP'en

- <%@ page isErrorPage="true" %>

Vi ser på et eksempel:

- http://matiksi.hive.no:8180/ thomas/Feilmelding.jsp

`http://matiksi.hive.no:8180/~thomas/Feilmelding.jsp`

- eksempler/Feilmelding.jsp

Listing 10.6: eksempler/Feilmelding.jsp

```
1  <%@ page isErrorPage="true" %>
2
3
4  <HTML>
5  <HEAD>
6     <TITLE>JSP-feilmeldings-side</TITLE>
7  <PRE>
8
9  JSP-feilmeldings-side
10 ---------------------
11
12 Foelgende feil er oppstaatt:
13
14 <%=
15 exception.toString()
16 %>
17
18 </PRE>
19
20 </BODY>
21 </HTML>
```

- http://matiksi.hive.no:8180/ thomas/DBFailConnect.jsp

`http://matiksi.hive.no:8180/~thomas/DBFailConnect.jsp`

- eksempler/DBFailConnect.jsp

Listing 10.7: eksempler/DBFailConnect.jsp

```
1  <%@ page errorPage="Feilmelding.jsp" import="java.sql.*" %>
2
3  <HTML>
```

```
4  <HEAD>
5    <TITLE>DB-connect</TITLE>
6  </HEAD>
7  <BODY>
8
9  <%
10
11 Class.forName("com.mysql.jdbc.Driver");
12 Connection lnk = DriverManager.getConnection(
13   "jdbc:mysql://localhost/bokbase",
14   "dba",
15   "da"
16 );
17
18 ResultSet res;
19 res = lnk.createStatement().executeQuery("show tables");
20 %>
21
22 <PRE>
23
24 <% while(res.next())
25   out.println(res.getString(1));
26 %>
27
28 </PRE>
29 </BODY>
30 </HTML>
```

10.10 Øvelse

a)

- Gjør om SqlKlient2 fra et cgi-program til JSP.
- SqlKlient2.java

Listing 10.8: losningsforslag/SqlKlient2.java

```
1  import java.net.URLDecoder;
2  import java.sql.Connection;
3  import java.sql.DriverManager;
4  import java.sql.ResultSet;
5  import java.sql.ResultSetMetaData;
6  import java.sql.SQLException;
7  import java.sql.Statement;
8
9  public class SqlKlient2 {
10
11   private static ResultSetMetaData met;
12   private static Connection        lnk;
13   private static Statement         stm;
14   private static ResultSet         res;
15
16   public static void main(String[] args)
17   throws
18   SQLException,
```

```
19    ClassNotFoundException {
20
21      String url, drv, usr, pwd, sql;
22
23      url = "jdbc:mysql://localhost/bokbase";
24      drv = "com.mysql.jdbc.Driver";
25
26      usr = "db";
27      pwd = "ada";
28
29      Class.forName(drv);
30      lnk = DriverManager.getConnection(url,usr,pwd);
31      stm = lnk.createStatement();
32
33      try {
34        sql = URLDecoder
35        .decode(System.getenv("QUERY_STRING"), "UTF-8")
36        .substring(2);
37      } catch (Exception e) {
38        sql = "";
39      }
40
41      System.out.println("Content-type:text/html;charset=iso-8859-1\n\n");
42      System.out.printf(
43          "<form><input size=100 name=s value='%s' type=text></form>",
44          sql
45      );
46
47      sporring(sql);
48    }
49
50    private static void sporring(String sql) throws SQLException {
51
52      int kol;
53      int i;
54
55      res = stm.executeQuery(sql);
56      met = res.getMetaData();
57      kol = met.getColumnCount();
58      System.out.println("<table border=1>");
59
60      System.out.println("<tr>");
61      for (i=1; i<=kol; i++)
62        System.out.printf("<th>%s</th>", met.getColumnName(i));
63      System.out.println("</tr>");
64
65      while(res.next()){
66        System.out.println("<tr>");
67        for ( i = 1; i <= kol; i++ )
68          System.out.printf("<td>%s</td>",res.getString(i));
69        System.out.println("</tr>");
70      }
71
72      System.out.println("</table>");
73    }
74  }
```

b)

- Lag en egen feilmeldings-side til JSP'en i a).

c)

- Forandre Sesjonsdata.jsp slik at det også øker verdien 'teller' hver gang den blir forespurt.

- Sesjonsdata.jsp

Listing 10.9: losningsforslag/Sesjonsdata.jsp

```
1  <HTML>
2  <HEAD>
3    <TITLE>Sesjonsdata i JSP</TITLE>
4  </HEAD>
5  <BODY>
6
7
8  Sesjonens identifikator (cookie med navnet JSESSIONID):
9  <BR>
10 <%= session.getId()%>
11
12 <P>
13
14 Sesjonens Integer med navn 'teller':
15 <BR>
16 <%= session.getAttribute("teller") %>
17
18 <P>
19
20 <A HREF=SesjonsTest2> Hyperlenke til servlet </A>
21
22 </BODY>
23 </HTML>
```

Kapittel 11

JavaBeans

11.1 Løsning på tidligere oppgave

10 a)

- Gjør om SqlKlient2 fra et cgi-program til JSP.

10 b)

- Lag en egen feilmeldings-side til JSP'en i a).
- sqlklient.jsp

Listing 11.1: losningsforslag/sqlklient.jsp

```
1  <%@ page errorPage="Feilmelding.jsp" import="java.sql.*" %>
2
3  <HTML>
4  <HEAD>
5    <TITLE>DB-connect</TITLE>
6  </HEAD>
7  <BODY>
8
9  <%
10 Class.forName("com.mysql.jdbc.Driver");
11 Connection lnk = DriverManager.getConnection(
12   "jdbc:mysql://localhost/bokbase",
13   "db",
14   "ada"
15 );
16 %>
17
18 <form>
19  <input
20   type=text
21   size=100
22   name=s value='<%= request.getParameter("s")%>'
23  />
24 </form>
25 <table border=1>
26 <tr>
27 <%
```

```
28 ResultSetMetaData met;
29 ResultSet res;
30 int kol, i;
31
32 res = lnk.createStatement().executeQuery(request.getParameter("s"));
33 met = res.getMetaData();
34 kol = met.getColumnCount();
35 out.print(kol);
36
37  for (i=1; i <= kol; i++)
38    out.print(met.getColumnName(i));
39 %>
40 </tr>
41
42 <% while(res.next()){ %>
43
44  <tr>
45
46 <% for ( i = 1; i <= kol; i++ ) {%>
47
48   <td> <%= res.getString(i) %> </td>
49
50 <%   }%>
51 </tr>
52 <%}%>
53
54 </table>
55 </BODY>
56 </HTML>
```

11.2 JavaBeans

- Gjenbruk - generelle programmer som plugges inn i applikasjoner

Ved programvare-utvikling

- Lavere kostnad
- Kortere tid

annen velkjent komponentmodell - ActiveX

- 'ActiveX bridge' kan konvertere en JavaBean til ActiveX

En individuell komponent kalles ofte

- JavaBean
- bean
- i dette dokumentet kalt *bønne*

Eksempler på bønner

- Swing-komponenter
- AWT-komponenter

innkapsling

- Implementasjonsdetaljer kan være skjult for applikasjonsutvikleren.

For å bruke eksisterende bønne, trenger applikasjons-programmerer trenger kun å vite

- navn
- argumenter
- returverdi

Å lage en JavaBean

- må være i navngitt pakke
- vanligvis uten main()
- alle "ikke-biblioteksmetoder" bør begynne med "get-" eller "set-"
- *bør* implementere Serializable
- pakk eventuelt inn i jar-fil sammen med manifest-fil (påkrevd for bruk i IDE)

Eksempel

- *jar cmf Manifestfil.mf Bonne.jar *.class*

Minimal manifestfil (Manifest.mf):

```
Name: pakke/Bonne.class
Java-Bean: True
```

11.3 JavaBeans i JSP

- må ha en konstruktør uten argumenter
- *må* implementere Serializable
- kun getters og setters er tilgjengelige i JSP
- pakken (med klassefila) må kopieres inn i 'WEB-INF/classes'
- når bønna er på plass, kan den "getters" og "setters" brukes

må spesifiseres med en action-tag

- spesifiserer biblioteket 'jsp'
- "useBean" i navnet
- bønnenavn i 'id'
- pakke og klasse i 'class'

tre andre mulige attributter

- type
- beanName

'scope' angir tilgjengelighet

- page
- request
- session
- application

actiontag-egenskaper

- 'getters' og 'setters' er tilgjengelig via action-tags - spesielt kjekt for ikke-programmerere
- <jsp:setProperty>
- <jsp:getProperty>

to påkrevde attributter

- 'name' bønnas navn
- 'property' egenskapens navn

attributter for setProperty

- 'value'

Html-parametre

- Hvis parametre (f.eks. fra html-skjema), skal settes i bønna, finnes tre måter UTEN å bruke 'value'-attributtet

1.

- Hvis parameteret har samme navn og type som bønne-egenskapen, kan 'value' droppes
- <jsp:setProperty name="..." property="..."

2.

- Hvis parameteret har et annet navn: Bruk attrubuttet 'param' istedet
- <jsp:setProperty name="..." property="..." param="annetNavn"

3.

- Sett alle bønnas egenskaper til parameterverdier med matchende navn og type
- <jsp:setProperty name="..." property="*"

11.4 Eksempler

Hallo

- bonner/HalloBonne.java

Listing 11.2: eksempler/bonner/HalloBonne.java

```
1  package bonner;
2
3  public class HalloBonne implements java.io.Serializable {
4
5    public String getHilsen () {
6
7      return "Hallo";
8    }
9
10 }
```

- bonne.jsp

Listing 11.3: eksempler/bonne.jsp

```
1  <html>
2  <head><title>Hallo-bonne test </title></head>
3  <body>
4
5     <jsp:useBean id="bonne" class="bonner.HalloBonne"/>
6
7     <%= bonne.getHilsen() %>
8
9     <jsp:getProperty name="bonne" property="hilsen" />
10
11    ${bonne.hilsen}
12
```

```
13
14
15 </body>
16 </html>
```

- http://matiksi.hive.no:8180/ thomas/bonne.jsp

`http://matiksi.hive.no:8180/~thomas/bonne.jsp`

Person 1

- bonner/PersonBonne.java

Listing 11.4: eksempler/bonner/PersonBonne.java

```
1  package bonner;
2
3  public class PersonBonne
4  implements java.io.Serializable {
5
6    private String fornavn;
7    private String etternavn;
8    private int lonn;
9
10   public String getEtternavn() {
11     return etternavn;
12   }
13   public void setEtternavn(String etternavn) {
14     this.etternavn = etternavn;
15   }
16   public String getFornavn() {
17     return fornavn;
18   }
19   public void setFornavn(String fornavn) {
20     this.fornavn = fornavn;
21   }
22   public int getLonn() {
23     return lonn;
24   }
25   public void setLonn(int lonn) {
26     this.lonn = lonn;
27   }
28 }
```

- person1.jsp

Listing 11.5: eksempler/person1.jsp

```
1 <%@ page errorPage="Feilmelding.jsp" %>
2
3 <HTML>
4 <HEAD>
5   <TITLE> Person-skjema m/bonne </TITLE>
6 </HEAD>
7 <BODY>
8
```

```
<jsp:useBean id="pB" class="bonner.PersonBonne" />

<jsp:setProperty name="pB" property="fornavn" value="Thomas" />
<jsp:setProperty name="pB" property="etternavn" param="etternavn" />
<jsp:setProperty name="pB" property="lonn" />

<FORM method=post>
 Fornavn
 <INPUT
    type=text
    size=60
    name=fornavn
    value='<jsp:getProperty name="pB" property="fornavn" />'
 /> <BR>

 Etternavn
 <INPUT
    type=text
    size=60
    name=etternavn
    value='<%= pB.getEtternavn() %>'
 /> <BR>

 Lonn
 <INPUT
   type=text
   size=15
   name=lonn
   value='${pB.lonn}'
 /> <BR>

 <INPUT type=submit />
</FORM>

</BODY>
</HTML>
```

- http://matiksi.hive.no:8180/ thomas/person1.jsp

`http://matiksi.hive.no:8180/~thomas/person1.jsp`

Person 2

- person2.jsp

Listing 11.6: eksempler/person2.jsp

```
<%@ page errorPage="Feilmelding.jsp" %>

<HTML>
<HEAD>
  <TITLE> Person-skjema m/bonne </TITLE>
</HEAD>
<BODY>

<jsp:useBean id="pB" class="bonner.PersonBonne" />
```

```
10
11 <jsp:setProperty name="pB" property="*" />
12
13 <FORM method=post>
14  Fornavn
15  <INPUT
16     type=text
17     size=60
18     name=fornavn
19     value='<jsp:getProperty␣name="pB"␣property="fornavn"␣/>'
20  /> <BR>
21
22  Etternavn
23  <INPUT
24     type=text
25     size=60
26     name=etternavn
27     value='<%=␣pB.getEtternavn()␣%>'
28  /> <BR>
29
30  Lonn
31  <INPUT
32    type=text
33    size=15
34    name=lonn
35    value='${pB.lonn}'
36  /> <BR>
37
38  <INPUT type=submit />
39 </FORM>
40
41 </BODY>
42 </HTML>
```

- http://matiksi.hive.no:8180/ thomas/person2.jsp

`http://matiksi.hive.no:8180/~thomas/person2.jsp`

JDBC

- bonner/SqlBonne.java

Listing 11.7: eksempler/bonner/SqlBonne.java

```
1  package bonner;
2
3  public class SqlBonne implements java.io.Serializable {
4
5    private static java.sql.ResultSetMetaData met;
6    private static java.sql.Statement          stm;
7
8    private String sql = "show␣tables";
9
10   public SqlBonne() throws
11   java.sql.SQLException,
12   ClassNotFoundException {
```

```
13
14      Class.forName("com.mysql.jdbc.Driver");
15      stm = java.sql.DriverManager.getConnection(
16           "jdbc:mysql://localhost/bokbase",
17           "db",
18           "ada"
19      ).createStatement();
20    }
21
22    public void setSporring(String sporring){
23      sql = sporring;
24    }
25
26    public String getResultHtmlTable()
27    throws java.sql.SQLException {
28
29      String htmlTable;
30      java.sql.ResultSet res;
31      int kol, i;
32
33      res = stm.executeQuery(sql);
34      met = res.getMetaData();
35      kol = met.getColumnCount();
36
37      htmlTable = "<table><tr>";
38      for (i=1; i<=kol; i++)
39        htmlTable+="<th>" + met.getColumnName(i) + "</th>";
40      htmlTable+="</tr>";
41
42      while(res.next()){
43        htmlTable+="<tr>";
44        for ( i = 1; i <= kol; i++ )
45        htmlTable+="<td>" + res.getString(i) + "</td>";
46        htmlTable+="</tr>";
47      }
48
49      htmlTable+="</table>";
50
51      return htmlTable;
52    }
53  }
```

- sqlBonneKlient.jsp

Listing 11.8: eksempler/sqlBonneKlient.jsp

```
1  <%@ page errorPage="Feilmelding.jsp" %>
2
3  <HTML>
4  <HEAD>
5    <TITLE>SQL-bonne-klient</TITLE>
6  </HEAD>
7  <BODY>
8
9  <FORM>
10  <INPUT
11    type=text
12    size=100
```

```
13    name=sporring value='<%= request.getParameter("sporring") %>'
14   />
15  </FORM>
16
17  <jsp:useBean id="sqlBonne" class="bonner.SqlBonne" />
18  <jsp:setProperty name="sqlBonne" property="sporring" />
19  <jsp:getProperty name="sqlBonne" property="resultHtmlTable" />
20
21  </BODY>
22  </HTML>
```

- http://matiksi.hive.no:8180/ thomas/sqlBonneKlient.jsp

`http://matiksi.hive.no:8180/~thomas/sqlBonneKlient.jsp`

11.5 Øvelse

- Eksempelet med sqlklient-bønna returnerer et html-formatert tabell. Gjør den om slik at den blir mer generell, og dermed kan brukes til eventuelt andre format. Lag også en JSP som tar bønna i bruk. Tips: Se eksempel og oppgave i boka.

Kapittel 12

Applets og CORBA

12.1 Applets

- GUI-baserte javaprogrammer utformet for kjøring i HTML-sider, av nettlesere med innebygd JVM

Testing/kjøring av applet

- Applets kan (og bør) testes i nettlesere.
- Appletviewer er et program for å testing applet uten bruk av nettleser.
- Applets har ikke main() - styres av nettleser/appletviewer
- <applet code="..." codebase="..." width="..." height="..."></applet>
- med codebase kan katalog angis. Dette er påkrevet dersom katalogen med klassefilene ikke er en underkatalog av katalogen som html-filen ligger i.

Nettleser(/appletviewer) kaller tre metoder i denne rekkefølge

1. init()

- initialisering av variabler
- lasting av bilder
- etc.
- kalles ved klasse-lasting

2 start()

- starte animasjoner
- start tråder
- etc.
- kalles ved klasse-lasting etter init()

3. paint()

- tegne på pre-swing-applet
- kalles hver gang vinduet må om-tegnes

preSwing-applet

- extends java.applet.Applet

```
java.lang.Object
<|-- java.awt.Component
<|----java.awt.Container
<|------java.awt.Panel
<|-------java.applet.Applet
```

java.applet.Applet.paint()

- her skrives opptegningskode
- kalles av nettleser/appletviewer - ikke av applet'en
- metodene arves og kan overstyres

Eksempel:

- HalloApplet1.java

Listing 12.1: eksempler/HalloApplet1.java

```
1 public class HalloApplet1 extends java.applet.Applet {
2
3   public void paint (java.awt.Graphics g) {
4
5     g.drawString("Hallo", 10, 10);
6   }
7 }
```

- halloApplet1.html

Listing 12.2: eksempler/halloApplet1.html

```
1 <html>
2
3   HalloApplet1:
4   <hr>
5   <applet code="HalloApplet1.class" width="300" height="100">
6   </applet>
7
8 </html>
```

Swing-applet

- extends javax.swing.JApplet

```
java.lang.Object
<|-- java.awt.Component
<|----java.awt.Container
<|------java.awt.Panel
<|-------java.applet.Applet
<|---------javax.swing.JApplet
```

- Unngår direkte opptegning (painting)

javax.swing.JApplet.init()

- her legges komponenter til overflaten
- ingen opptegning spesifiseres
- kalles implisitt av nett-leser/appletviewer

Eksempel:

- HalloApplet2.java

Listing 12.3: eksempler/HalloApplet2.java

```
1 public class HalloApplet2 extends javax.swing.JApplet {
2
3   public void init () {
4
5     add(new javax.swing.JLabel("Hallo"));
6   }
7 }
```

- halloApplet2.html

Listing 12.4: eksempler/halloApplet2.html

```
1 <html>
2   HalloApplet2:
3   <hr>
4   <applet code="HalloApplet2.class" width="300" height="100">
5   </applet>
6 </html>
```

Eksempel på interaktiv applet

- InteraktivApplet.java

Listing 12.5: eksempler/InteraktivApplet.java

```
1
2  public class InteraktivApplet extends javax.swing.JApplet {
3
4    String hilsen;
5
6    public void init () {
7
8      hilsen = javax.swing.JOptionPane.showInputDialog(
9          "Skriv din hilsen her");
10
11     add(new javax.swing.JLabel(hilsen));
12
13   }
14
15 }
```

- interaktivApplet.html

Listing 12.6: eksempler/interaktivApplet.html

```
1 <html>
2   InteraktivApplet:
3   <hr>
4   <applet code="InteraktivApplet.class" width="300" height="100">
5   </applet>
6   <p>
7
8 </html>
```

Bilder

- opprinnelig kun støtte for gif
- JDK 1.1. - kom støtte for JPEG (.jpg/.jpeg)

java.awt.Image

```
java.lang.Object
<|-- java.awt.Image
```

- Abstract: Kan lage referanse - ikke instans direkte
- plattformavhengig subklasse utilgjengelig for applikasjons-programmerere

Applet-getImage()

- returnerer ref. til nedlastet bilde. (som ref.til Image)
- kjører i egen tråd
- Graphics.drawImage() kan brukes til å vise bildet i applet

bildeApplet1

- eksempler/bildeApplet1.html

Listing 12.7: eksempler/bildeApplet1.html

```
<html>
  BildeApplet1:
  <hr>
  <applet code="BildeApplet1.class" width="300" height="100">
  </applet>
</html>
```

- http://matiksi.hive.no/oopva60/eksempler/bildeApplet1.html

`http://matiksi.hive.no/oopva60/eksempler/bildeApplet1.html`

- eksempler/BildeApplet1.java

Listing 12.8: eksempler/BildeApplet1.java

```
public class BildeApplet1 extends java.applet.Applet {

  java.awt.Image bilde;

  public void init() {

    bilde = getImage(getDocumentBase(), "logo_HVE_rod.jpg");
  }

  public void paint (java.awt.Graphics g) {

    g.drawImage(bilde, 0, 0, this);
  }
}
```

javax.swing.ImageIcon

```
java.lang.Object
<|-- javax.swing.ImageIcon
```

spesielt hendig ved

- lasting av blider fra aktiv katalog
- overføring av blider ('implements Serializable') – kan sendes som objektstrøm
- kan brukes i konstruktører for JLables og JButtons - det kan ikke Images

Eksempler

bildeApplet2

- eksempler/bildeApplet2.html

Listing 12.9: eksempler/bildeApplet2.html

```
<html>
  BildeApplet2:
  <hr>
  <applet code="BildeApplet2.class" width="300" height="100">
  </applet>
</html>
```

- http://matiksi.hive.no/oopva60/eksempler/bildeApplet2.html

`http://matiksi.hive.no/oopva60/eksempler/bildeApplet2.html`

- eksempler/BildeApplet2.java

Listing 12.10: eksempler/BildeApplet2.java

```
public class BildeApplet2 extends javax.swing.JApplet {

  javax.swing.ImageIcon bilde;

  public void init() {

    bilde = new javax.swing.ImageIcon(
        getDocumentBase(),
        "logo_HVE_rod.jpg"
        );
  }

  public void paint (java.awt.Graphics g) {

    bilde.paintIcon(this, g, 0, 0);
  }
}
```

bildeApplet2

- eksempler/bildeApplet3.html

Listing 12.11: eksempler/bildeApplet3.html

```
<html>
  BildeApplet2:
  <hr>
  <applet code="BildeApplet3.class" width="300" height="100">
  </applet>
</html>
```

- http://matiksi.hive.no/oopva60/eksempler/bildeApplet3.html

`http://matiksi.hive.no/oopva60/eksempler/bildeApplet3.html`

- eksempler/BildeApplet3.java

Listing 12.12: eksempler/BildeApplet3.java

```
1  public class BildeApplet3 extends javax.swing.JApplet {
2
3    javax.swing.ImageIcon bilde;
4
5    public void init() {
6
7      bilde = new javax.swing.ImageIcon(
8          getImage(
9              getDocumentBase(),
10             "logo_HVE_rod.jpg"
11         )
12     );
13   }
14
15   public void paint (java.awt.Graphics g) {
16
17     bilde.paintIcon(this, g, 0, 0);
18   }
19 }
```

Beregning av relativ størrelse

- Component.getHeight()
- Component.getWidth()

Lyd

først kun støtte for Sun Audio format (.au) senere

- Windows wave format (.wav)
- Macintosh AIFF (.aif)
- MIDI (.mid/.rmf)

avspilling

java.applet.Applet.play()

- For å spille av lyden en gang

java.applet.AudioClip.play()

- For applikasjoner

For å kunne spille av lyden flere ganger

- AudioClip.play()
- AudioClip.stop()
- AudioClip.loop()
- java.applet.AudioClip.play(URL, fil)
- java.applet.AudioClip.play(URLl)
- Ved å unngå å måtte omgå appletens sikkerhets-restriksjoner, kan applet og lydfil holdes i samme katalog

eksempler

lydApplet1

- eksempler/lydApplet1.html

Listing 12.13: eksempler/lydApplet1.html

```
<html>
  LydApplet1:
  <hr>
  <applet code="LydApplet1.class" width="300" height="100">
  </applet>
</html>
```

- http://matiksi.hive.no/oopva60/eksempler/lydApplet1.html

`http://matiksi.hive.no/oopva60/eksempler/lydApplet1.html`

- eksempler/LydApplet1.java

Listing 12.14: eksempler/LydApplet1.java

```
public class LydApplet1

extends java.applet.Applet {

  public void init(){

    try {
      play(
          new java.net.URL(
              getDocumentBase(),
              "Tuxedomoon.Jingle4.wav"
          )
      );

    } catch (java.net.MalformedURLException e) {

      e.printStackTrace();
    }
  }
}
```

lydApplet2

- eksempler/lydApplet2.html

Listing 12.15: eksempler/lydApplet2.html

```
<html>
  LydApplet2:
  <hr>
  <applet code="LydApplet2.class" width="300" height="100">
  </applet>
</html>
```

- http://matiksi.hive.no/oopva60/eksempler/lydApplet2.html

`http://matiksi.hive.no/oopva60/eksempler/lydApplet2.html`

- eksempler/LydApplet2.java

Listing 12.16: eksempler/LydApplet2.java

```
public class LydApplet2
extends javax.swing.JApplet
implements java.awt.event.ActionListener {

  javax.swing.JButton spill, stopp, loop;
  javax.swing.JPanel panel;
  java.applet.AudioClip lyd;


  public void init(){

    try {

      lyd = getAudioClip(
          new java.net.URL(
              getDocumentBase(),
              "Tuxedomoon.Jingle4.wav"
          )
      );

    } catch (java.net.MalformedURLException e) {

      e.printStackTrace();
    }

    spill = new javax.swing.JButton("Spill");
    stopp = new javax.swing.JButton("Stopp");
    loop  = new javax.swing.JButton("Loop");

    spill.addActionListener(this);
    stopp.addActionListener(this);
    loop. addActionListener(this);

    panel = new javax.swing.JPanel();

    panel.add(spill);
    panel.add(stopp);
```

```
38      panel.add(loop);
39
40      this.add(panel);
41    }
42
43    public void actionPerformed(
44        java.awt.event.ActionEvent e) {
45
46      Object o = e.getSource();
47
48      if (o == spill)
49        lyd.play();
50
51      else if (o == stopp)
52        lyd.stop();
53
54      else if (o == loop)
55        lyd.loop();
56    }
57  }
```

12.2 CORBA

- Common Object Request Broker Architecture (CORBA)
- For objekter laget i ulike språk
- En implementasjon av spesifikasjonen kalles en Object Request Broker (ORB)
- eks. på en ORB: Java IDL
- CORBA-ORB:Internet Inter-Orb Protorcol (IIOP) - basert på TCP/IP: Gir interoperabilitet mellom ulike leverandører

For hver fjern-metode er det en

- ”stub” på klientsiden
- ”skeleton” på tjenersiden

IDL

- ”Interface’ene” defineres i med Interface Definition Language (IDL)
- ikke fullstendig språk
- ’C++’-lignende
- ORB må oversette fra IDL til målspråket

Det finnes oversettere til

- java
- C++
- C
- SmallTalk
- COBOL
- Ada

12.3 Øvelser

11.1

- a) Skriv kort om forskjeller og likheter mellom applets og applikasjoner.
- b) Skriv kort om forskjeller og likheter mellom applets og servlets.

11.3.

- Skriv om løsningsforlaget til 10 a/b fra JSP til applet.
- losningsforslag/sqlklient.jsp

Listing 12.17: losningsforslag/sqlklient.jsp

```
1  <%@ page errorPage="Feilmelding.jsp" import="java.sql.*" %>
2
3  <HTML>
4  <HEAD>
5    <TITLE>DB-connect</TITLE>
6  </HEAD>
7  <BODY>
8
9  <%
10 Class.forName("com.mysql.jdbc.Driver");
11 Connection lnk = DriverManager.getConnection(
12   "jdbc:mysql://localhost/bokbase",
13   "db",
14   "ada"
15 );
16 %>
17
18 <form>
19  <input
20   type=text
21   size=100
22   name=s value='<%= request.getParameter("s")%>'
23  />
24 </form>
```

```
25 <table border=1>
26 <tr>
27 <%
28 ResultSetMetaData met;
29 ResultSet res;
30 int kol, i;
31
32 res = lnk.createStatement().executeQuery(request.getParameter("s"));
33 met = res.getMetaData();
34 kol = met.getColumnCount();
35 out.print(kol);
36
37  for (i=1; i <= kol; i++)
38    out.print(met.getColumnName(i));
39 %>
40 </tr>
41
42 <% while(res.next()){ %>
43
44  <tr>
45
46 <% for ( i = 1; i <= kol; i++ ) {%>
47
48   <td> <%= res.getString(i) %> </td>
49
50 <%   }%>
51 </tr>
52 <%}%>
53
54 </table>
55 </BODY>
56 </HTML>
```

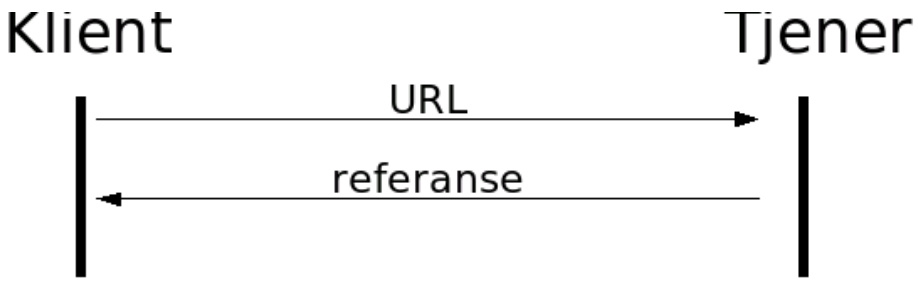
Kapittel 13

Oppsummering del 2

13.1 Remote Method Invocation

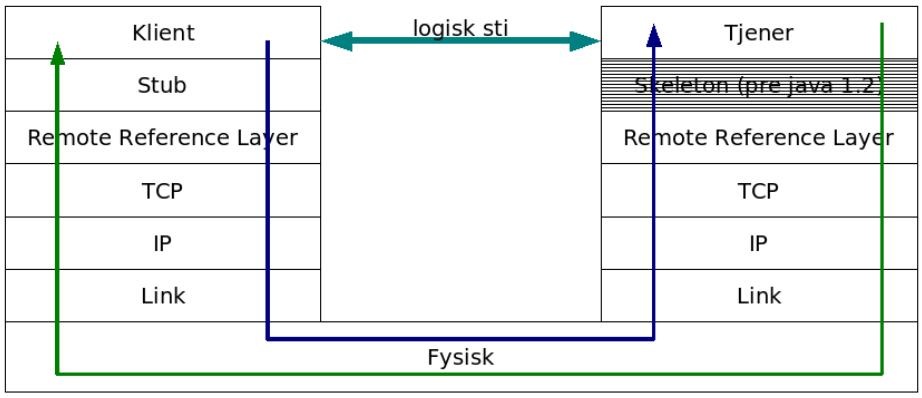
- Tjener registrerer et grensesnitt hos en navnetjener. Dette grensesnittet inneholder signaturene til de metodene som skal tilbys.
- Klienten gjør et navneoppslag i et register på en URL, og får en referanse tilbake.

figur



- Når klientene forespør tjenesten får den en referanse til et *remote object* i form av en *stub*. Et remote object implementerer et *remote interface*.
- Stubben videresender metodekall og parametre til en skeleton på det fjerne systemet.

figur



Parametre:

- Verdioverføring: Primitive typer.
- Referanseoverføring: Serialiserbare objekter (implements serializable) blir serialisert og kopiert (som verdioverføring).

RMISecurityManager

- extends SecurityManager
- forekomst kontrollerer implementasjon av Security Policy
- uten et slikt objekt vil ikke klasser lastes hvis de ikke er på lokalt filsystem

Tjener

Interface

- Et remote object er definert som: En forekomst av en klasse som implementerer remote interface, eller et interface som extends Remote.
- Interface'et bestemmer hvilke metoder som er tilgjengelige utenfra.
- Hallogrensesnitt.java

Listing 13.1: eksempler/Hallogrensesnitt.java

```
1 public interface Hallogrensesnitt extends java.rmi.Remote{
2
3   public String hallo() throws java.rmi.RemoteException;
4
5 }
```

Implementasjon av Interface

- FjernHallo.java

Listing 13.2: eksempler/FjernHallo.java

```
1  public class FjernHallo
2  extends java.rmi.server.UnicastRemoteObject
3  implements Hallogrensesnitt{
4
5    public FjernHallo() throws java.rmi.RemoteException {
6
7    }
8
9    public String hallo() throws java.rmi.RemoteException {
10
11     return "Hallo";
12   }
13 }
```

Tjener-klasse

- Tjeneren må gjøre to ting
- 1. Opprette objekt av klassen som implementerer interfacet
- 2. Knytte objekte til et navn (med Naming.bind() eller Naming.rebind()).
- Klienter kan spørre med navn eller etter en liste over tilgjengelige metoder.
- Det kan være flere instanser av samme klasse som er registrert med ulike navn. main() returnerer raskt, men tjeneren forstetter å kjøre...
- Registeret må startes før objektet kan knyttes til det. Dette kan gjøres med kommandoen 'rmiregistry&', eller i direkte i tjenerklassen.
- RmiTjener3.java

Listing 13.3: eksempler/RmiTjener3.java

```
1  public class RmiTjener3 {
2
3    public static void main (String[] args) throws
4    java.net.MalformedURLException,
5    java.rmi.RemoteException{
6
7      System.setSecurityManager(new java.rmi.RMISecurityManager());
8
9      // alternativ til aa starte navneregisteret fra kommandolinjen (
           rmiregistry &)
10     java.rmi.registry.LocateRegistry.createRegistry(1099);
11
12     java.rmi.Naming.rebind(
13         "Hallo",
14         new FjernHallo()
15     );
16   }
17 }
```

- Banen til sikkerhetspolicy-fila kan angis som kommandolinje-argument (*-Djava.security.policy=tje*
- tjener.policy

Listing 13.4: eksempler/tjener.policy

```
1  grant {
2    permission java.net.SocketPermission "matiksi.hive.no", "connect,resolve"
         ;
3    permission java.net.SocketPermission "*", "accept";
4  };
```

Klient

- For at et objekt skal kunne kalle på en metode i et fjernt objekt, må det ha en "remote reference" til det fjerne objektet. Klienten skaffer seg dette, ved henvendelse til registeret på vertsmaskinen hvor det fjerne objektet finnes. Henvendelsen gjøres ved å kalle på registerets lookup()-metode.

- Objekt-referansen som returneres må kastes om til interfacet (klassen er ikke synlig for klienten). Objekter mister info om type når de lagres i strukturer som kan inneholde objekter av ulike klasser.

- URL'ene er bygget opp på samme måte som http-URL'er: rmi://vert.domene:port/navn

- Det fjerne objektet kan nå brukes som et lokalt. Den eneste forskjellen er at "RemoteExeption" må fanges ved bruk).

- Kompiler som vanlig - du trenger byte- /kilde- koden for interfacet (ikke klassen som implementerer den) tilgjengelig i klassestien.

- Banen til sikkerhetspolicy-fila kan angis som kommandolinje-argument (*-Djava.security.policy=kli*

- RmiKlient2.java

Listing 13.5: eksempler/RmiKlient2.java

```
1  public class RmiKlient2 {
2
3    public static void main(String[] args) throws
4    java.net.MalformedURLException,
5    java.rmi.NotBoundException,
6    java.rmi.RemoteException
7    {
8      // For aa tillate kjoering av nedlastet kode
9      System.setSecurityManager(new java.rmi.RMISecurityManager());
10
11     Hallogrensesnitt h =
12       (Hallogrensesnitt)java.rmi.Naming.lookup(
13           "rmi://matiksi.hive.no/Hallo");
14
15     System.out.println(h.hallo());
16   }
17
18 }
```

- klient.policy

Listing 13.6: eksempler/klient.policy

```
1 grant {
2     permission java.net.SocketPermission "matiksi.hive.no", "connect,
          resolve";
3 };
```

13.2 Servlets

Om servlets

- En *servlet* er et program skrevet i java som kjører på en web-tjener. Dette står i motsetning til en *applet* som kjøres på klienten.

Servlets container

- The container is the intermediary between the Web server and the servlets in the container. The container loads, initializes, and executes the servlets.
- Kjøringen trigges av http-forespørsel fra web-klient.
- Produserer dokument (f.eks. et html-dokument) som returneres til klienten.

Eksempler

Eksempel 2: Input fra html-skjema

- eksempler/serveline.html

Listing 13.7: eksempler/serveline.html

```
<html>
<form action="http://matiksi.hive.no:8180/~thomas/Serveline">
Brukernavn: <input type=text name=Brukernavn> <p>
<input type=submit>
</form>
</html>
```

- Serveline.java

Listing 13.8: eksempler/Serveline.java

```
public class Serveline  extends javax.servlet.http.HttpServlet {

    public void doGet(
        javax.servlet.http.HttpServletRequest  request,
        javax.servlet.http.HttpServletResponse respons )

        throws java.io.IOException,
               javax.servlet.ServletException {
      java.io.PrintWriter ut = respons.getWriter();

        String brukernavn = request.getParameter("Brukernavn");


        respons.setContentType("text/Plain");
        ut.println("Hallo "+brukernavn+", jeg heter Serveline");

        ut.flush();
    }
}
```

Eksempel 3: Informasjonskapsler

- http://matiksi.hive.no:8180/ thomas/CookieTest

`http://matiksi.hive.no:8180/~thomas/CookieTest`

- For å se informasjonskapslene i firefox: Edit preferences -> privacy -> show cookies
- CookieTest.java

Listing 13.9: eksempler/CookieTest.java

```
1  public class CookieTest  extends javax.servlet.http.HttpServlet {
2
3      public void doGet(
4          javax.servlet.http.HttpServletRequest  request,
5          javax.servlet.http.HttpServletResponse respons )
6
7          throws java.io.IOException,
8                 javax.servlet.ServletException {
9
10       java.io.PrintWriter ut;
11       javax.servlet.http.Cookie nybaktKake;
12       javax.servlet.http.Cookie[] kake;
13       int i;
14
15       ut = respons.getWriter();
16       respons.setContentType("text/html");
17
18       kake = request.getCookies();
19       if (kake == null) {
20
21         nybaktKake = new javax.servlet.http.Cookie("IP", request.
               getRemoteAddr() );
22           nybaktKake.setMaxAge(60*60);
23           respons.addCookie(nybaktKake);
24           ut.println("Velkommen␣fremmede");
25
26       } else {
27         // skriver ut verdiene i informasjonskapslene
28         ut.println("Takk␣for␣sist!");
29         for (i=0; i<kake.length;i++)
30           ut.printf(
31                "%s:␣%s<br>",
32                kake[i].getName(),
33                kake[i].getValue()
34           );
35       }
36
37         ut.flush();
38     }
39 }
```

Eksempel 4: Sesjoner

- SesjonsTest.java

Listing 13.10: eksempler/SesjonsTest.java

```
1  public class SesjonsTest  extends javax.servlet.http.HttpServlet {
2
3    public void doGet(
4        javax.servlet.http.HttpServletRequest  request,
5        javax.servlet.http.HttpServletResponse respons )
6
7    throws
8    java.io.IOException,
9    javax.servlet.ServletException {
10
11     java.io.PrintWriter ut;
12
13     javax.servlet.http.HttpSession session = request.getSession();
14
15     Integer ival = (Integer) session.getAttribute("sessiontest.counter");
16
17     if (ival==null)
18       ival = new Integer(1);
19     else
20       ival = new Integer(ival.intValue() + 1);
21     session.setAttribute("sessiontest.counter", ival);
22
23     ut = respons.getWriter();
24     respons.setContentType("text/html");
25
26     ut.println("<html><body>");
27     ut.println("Tellerverid: " + ival + "<p>");
28
29     if (session.isNew())
30       ut.println("Ny sesjon <p>SesjonsID:" + session.getId());
31     else
32       ut.println("Gammel sesjon.<p>SesjonsID:" + session.getId());
33
34
35         ut.println("<form><input type=submit></form></body></html>");
36     ut.flush();
37   }
38 }
```

- SesjonsTest

`http://matiksi.hive.no:8180/%7Ethomas/SesjonsTest`

- java.sun.com > Products > Servlet > 2.2 > Javadoc > Javax > Servlet > Http > HttpSession

`http://java.sun.com/products/servlet/2.2/javadoc/javax/servlet/http/HttpSession.html`

13.3 JavaServer Pages

Hva er JSP?

- HTML-dokumenter med innfelt javakode.

- Som en utvidelse av servlet-teknologien.
- JSP er en annen måte å skrive servlets uten å være "java-ekspert"
- fil-endelse: .jsp
- med JSP menes både selve web-sidene som lages og teknologien som brukes for å lage web-sidene

Eksempel:

- http://matiksi.hive.no:8180/ thomas/Hallo.jsp

`http://matiksi.hive.no:8180/~thomas/Hallo.jsp`

- eksempler/Hallo.jsp

Listing 13.11: eksempler/Hallo.jsp

```
1  <html>
2  <head>
3    <title>JSP-Hallo</title>
4  </head>
5  <body>
6
7    <%-- Dette er en kommentar --%>
8
9    <% out.println("Hallo"); %>
10
11 </body>
12 </html>
```

Når brukes JSP, og når brukes servlets?

- Programmer som ikke gir noe utskrift er gode kandiater til servlets

Servlets

- I servlets trengs kompentanse i javaprogrammering overalt
- F.eks ved nytt utseende, må javaprogrammet endres
- Vanskelig å dra nytte av webutviklings-verktøy
- Servlets bra for programmerere

kompilering/kjøring

- En web-tjener trenger en servlet-kontainer for å håndtere servlets, og en JSP-kontainer å håndtere JSP'er
- JSP-kontaineren avbryter forespørsel til JSPer og produserer servlet ved behov og sender forespørsel til servlet som er "mappet" til JSP'en.
- JSP-filene plasseres sammen med html-filene.
- JSP blir kompilert til servlets ved første klientforespørsel.
- Den kompilerte filen beholdes inntil tjeneren terminerer eller JSP-kildekoden endres.
- Dette fører til lang respostide ved første referanse.
- Prekompilering kan gjøres.
- Ved forespørsel blir template-tekst og JSP-elementer slått sammen

JElementer

To typer elementer

template text

- Blir alltid sendt rett gjennom til nettleseren

Statisk innhold

- HTML
- plain-text
- XML
- etc.

JSP-elementer

- noen få JSP-elementer for dynamisk innhold

Kategorier

- 1. Direktiver
- 2. Deklarasjoner
- 3. Uttrykk / "expressions"
- 4. Scriptlets
- 5. Kommentarer
- 6. Handling (action)
- (7. "Expression Language"-uttrykk)

1. Direktiver

- Informasjon om selve siden (som forblir lik mellom forespørsler)

<%@ page ... %>

- Attributter tilhørende selve siden. F.eks:
- contentType (del av http-header)
- import (samme som i java)
- errorPage (se eksempel under)

<%@ include ... %>

- Setter inn en fil i løpet av oversettelsesfasen

<%@ taglib ... %>

- spesifiserer et "tag library"

2. Deklarasjoner

- <%! ... %>
- Eks: <%! int teller; %>
- Dette produserer instansvariabler til servlet-klassen, som kan brukes i etterfølgende JSP-tagger.

3. Uttrykk / "expressions"

- kodeuttrykk hvis resultat skal med i respons. eller ved forspørsel brukt om "action-attributt-verdi"
- <%= ... %>
- Eks.: <%= teller++ %>
- Legg merke til at det ikke er semikolon i uttrykket.

4. Scriptlets

- <% ... vanlig java kode ... %>
- Ikke bruk dette mye - ved behov for mye kode, skill ut i servlet eller java-bean (se neste forelsning).
- Variabel-deklarasjoner kan også gjøres i scriptlets.
- Variabler deklarert i scriptlets, kan også brukes i etterfølgende JSP-tags.

5. Kommentarer

- <%– ... –%>
- Kommentarer fjernes, slik at de ikke sendes til klientene.

6. Handling (action)

- Ufører funksjoner som utgjør en utvidelse av JSP-standarden, f.eks. v.h.a. Java Beans
- Åpnings taggen spesifiserer bibliotek- og handlings-navn, adskilt med kolon:
- Eksempel: <jsp:useBean>
- Vi skal se mer om dette i forelesningen om java-beans.

(7. "Expression Language"-uttrykk)

- ikke pensum

Et eksempel med kommentarer, direktiver og uttrykk:

- http://matiksi.hive.no:8180/ thomas/Klokka.jsp

`http://matiksi.hive.no:8180/~thomas/Klokka.jsp`

- eksempler/Klokka.jsp

Listing 13.12: eksempler/Klokka.jsp

```
1  <html>
2  <head>
3    <title>Direktiver</title>
4  </head>
5  <body>
6
7
8  <%-- Direktiv --%>
9  <%@ page import="java.util.Calendar" %>
10
11 <%-- Expressions --%>
12
13 Klokka er
14
15 <%= Calendar.getInstance().get(Calendar.MINUTE) %>
16
17 over
18
19 <%= Calendar.getInstance().get(Calendar.HOUR) %>
20
21 </body>
22 </html>
```

Implisitte JSP-objekter

- En web-tjener med JSP-støtte skal gi JSP'ene tilgang til noen ferdig deklarerte objekter som er instanser av klasser definert i servlet- og JSP- spesifkasjonen.

F.eks.

- HttpServletRequest request
- HttpServletResponse response
- HttpSession session
- JspWriter out
- Throwable exception

Error pages

- Bruke servlet til å generere en feilmeldings-side og redirigere kontrollen til denne

eller bruke JSP-spesifisert måte å håndtere feil

- <%@ page errorPage=" ....jsp " %>

i feilmeldings-JSP'en

- <%@ page isErrorPage="true" %>

Vi ser på et eksempel:

- http://matiksi.hive.no:8180/ thomas/Feilmelding.jsp

`http://matiksi.hive.no:8180/~thomas/Feilmelding.jsp`

- eksempler/Feilmelding.jsp

Listing 13.13: eksempler/Feilmelding.jsp

```
1  <%@ page isErrorPage="true" %>
2
3
4  <HTML>
5  <HEAD>
6     <TITLE>JSP-feilmeldings-side</TITLE>
7  <PRE>
8
9  JSP-feilmeldings-side
10 ---------------------
11
12 Foelgende feil er oppstaatt:
13
14 <%=
15 exception.toString()
16 %>
```

```
17
18 </PRE>
19
20 </BODY>
21 </HTML>
```

13.4 JavaBeans

JavaBeans

- Gjenbruk - generelle programmer som plugges inn i applikasjoner

En individuell komponent kalles ofte

- JavaBean
- bean
- i dette dokumentet kalt *bønne*

Eksempler på bønner

- Swing-komponenter
- AWT-komponenter

innkapsling

- Implementasjonsdetaljer kan være skjult for applikasjonsutvikleren.

For å bruke eksisterende bønne, trenger applikasjons-programmerer trenger kun å vite

- navn
- argumenter
- returverdi

Å lage en JavaBean

- må være i navngitt pakke
- vanligvis uten main()
- alle "ikke-biblioteksmetoder" bør begynne med "get-" eller "set-"
- *bør* implementere Serializable
- pakk eventuelt inn i jar-fil sammen med manifest-fil (påkrevd for bruk i IDE)

Eksempel

- *jar cmf Manifestfil.mf Bonne.jar *.class*

Minimal manifestfil (Manifest.mf):

```
Name: pakke/Bonne.class
Java-Bean: True
```

JavaBeans i JSP

- må ha en konstruktør uten argumenter
- *må* implementere Serializable
- kun getters og setters er tilgjengelige i JSP
- pakken (med klassefila) må kopieres inn i 'WEB-INF/classes'
- må spesifiseres med en action-tag
- når bønna er på plass, kan den "getters" og "setters" brukes
- <jsp:setProperty name="bonnenavn" property="egenskapsnavn" value="verdi">
- <jsp:getProperty name="bonnenavn" property="egenskapsnavn">

Html-parametre

- Hvis parametre (f.eks. fra html-skjema), skal settes i bønna, finnes tre måter UTEN å bruke 'value'-attributtet

1.

- Hvis parameteret har samme navn og type som bønne-egenskapen, kan 'value' droppes
- <jsp:setProperty name="..." property="..."

2.

- Hvis parameteret har et annet navn: Bruk attrubuttet 'param' istedet
- <jsp:setProperty name="..." property="..." param="annetNavn"

3.

- Sett alle bønnas egenskaper til parameterverdier med matchende navn og type
- <jsp:setProperty name="..." property="*"

Eksempel

- bonner/PersonBonne.java

Listing 13.14: eksempler/bonner/PersonBonne.java

```
1  package bonner;
2
3  public class PersonBonne
4  implements java.io.Serializable {
5
6    private String fornavn;
7    private String etternavn;
8    private int lonn;
9
10   public String getEtternavn() {
11     return etternavn;
12   }
13   public void setEtternavn(String etternavn) {
14     this.etternavn = etternavn;
15   }
16   public String getFornavn() {
17     return fornavn;
18   }
19   public void setFornavn(String fornavn) {
20     this.fornavn = fornavn;
21   }
22   public int getLonn() {
23     return lonn;
24   }
25   public void setLonn(int lonn) {
26     this.lonn = lonn;
27   }
28 }
```

- person1.jsp

Listing 13.15: eksempler/person1.jsp

```
1  <%@ page errorPage="Feilmelding.jsp" %>
2
3  <HTML>
4  <HEAD>
5    <TITLE> Person-skjema m/bonne </TITLE>
6  </HEAD>
7  <BODY>
8
9  <jsp:useBean id="pB" class="bonner.PersonBonne" />
10
11 <jsp:setProperty name="pB" property="fornavn" value="Thomas" />
12 <jsp:setProperty name="pB" property="etternavn" param="etternavn" />
13 <jsp:setProperty name="pB" property="lonn" />
14
15 <FORM method=post>
16  Fornavn
17  <INPUT
18    type=text
```

```
19      size=60
20      name=fornavn
21      value='<jsp:getProperty␣name="pB"␣property="fornavn"␣/>'
22    /> <BR>
23
24    Etternavn
25    <INPUT
26      type=text
27      size=60
28      name=etternavn
29      value='<%=␣pB.getEtternavn()␣%>'
30    /> <BR>
31
32    Lonn
33    <INPUT
34      type=text
35      size=15
36      name=lonn
37      value='${pB.lonn}'
38    /> <BR>
39
40    <INPUT type=submit />
41  </FORM>
42
43  </BODY>
44  </HTML>
```

- http://matiksi.hive.no:8180/ thomas/person1.jsp

`http://matiksi.hive.no:8180/~thomas/person1.jsp`

13.5 Applets

- GUI-baserte javaprogrammer utformet for kjøring i HTML-sider, av nettlesere med innebygd JVM

Testing/kjøring av applet

- Applets kan (og bør) testes i nettlesere.
- Appletviewer er et program for å testing applet uten bruk av nettleser.
- Applets har ikke main() - styres av nettleser/appletviewer
- <applet code="..." codebase="..." width="..." height="..."></applet>
- med codebase kan katalog angis. Dette er påkrevet dersom katalogen med klassefilene ikke er en underkatalog av katalogen som html-filen ligger i.

Nettleser(/appletviewer) kaller tre metoder i denne rekkefølge

1. init()

- initialisering av variabler
- lasting av bilder
- etc.
- kalles ved klasse-lasting

2 start()

- starte animasjoner
- start tråder
- etc.
- kalles ved klasse-lasting etter init()

3. paint()

- tegne på pre-swing-applet
- kalles hver gang vinduet må om-tegnes

preSwing-applet

- extends java.applet.Applet

```
java.lang.Object
<|-- java.awt.Component
<|----java.awt.Container
<|------java.awt.Panel
<|-------java.applet.Applet
```

java.applet.Applet.paint()

- her skrives opptegningskode
- kalles av nettleser/appletviewer - ikke av applet'en
- metodene arves og kan overstyres

Swing-applet

- extends javax.swing.JApplet

```
java.lang.Object
<|-- java.awt.Component
<|----java.awt.Container
<|------java.awt.Panel
<|-------java.applet.Applet
<|---------javax.swing.JApplet
```

- Unngår direkte opptegning (painting)

javax.swing.JApplet.init()

- her legges komponenter til overflaten
- ingen opptegning spesifiseres
- kalles implisitt av nett-leser/appletviewer

Eksempel på interaktiv applet

- InteraktivApplet.java

Listing 13.16: eksempler/InteraktivApplet.java

```
1
2  public class InteraktivApplet extends javax.swing.JApplet {
3
4    String hilsen;
5
6    public void init () {
7
8      hilsen = javax.swing.JOptionPane.showInputDialog(
9          "Skriv din hilsen her");
10
11     add(new javax.swing.JLabel(hilsen));
12
13   }
14
15 }
```

- interaktivApplet.html

Listing 13.17: eksempler/interaktivApplet.html

```
1 <html>
2   InteraktivApplet:
3   <hr>
4   <applet code="InteraktivApplet.class" width="300" height="100">
5   </applet>
6   <p>
7
8 </html>
```

Bilder

- opprinnelig kun støtte for gif
- JDK 1.1. - kom støtte for JPEG (.jpg/.jpeg)

java.awt.Image

```
java.lang.Object
<|-- java.awt.Image
```

- Abstract: Kan lage referanse - ikke instans direkte
- plattformavhengig subklasse utilgjengelig for applikasjons-programmerere

Applet-getImage()

- returnerer ref. til nedlastet bilde. (som ref.til Image)
- kjører i egen tråd
- Graphics.drawImage() kan brukes til å vise bildet i applet

javax.swing.ImageIcon

```
java.lang.Object
<|-- javax.swing.ImageIcon
```

spesielt hendig ved

- lasting av blider fra aktiv katalog
- overføring av blider ('implements Serializable') – kan sendes som objektstrøm
- kan brukes i konstruktører for JLables og JButtons - det kan ikke Images

Lyd

først kun støtte for Sun Audio format (.au) senere

- Windows wave format (.wav)
- Macintosh AIFF (.aif)
- MIDI (.mid/.rmf)

avspilling

java.applet.Applet.play()

- For å spille av lyden en gang

java.applet.AudioClip.play()

- For applikasjoner

For å kunne spille av lyden flere ganger

- AudioClip.play()
- AudioClip.stop()
- AudioClip.loop()
- java.applet.AudioClip.play(URL, fil)
- java.applet.AudioClip.play(URLl)
- Ved å unngå å måtte omgå appletens sikkerhets-restriksjoner, kan applet og lydfil holdes i samme katalog

Eksempel

- eksempler/lydApplet2.html

Listing 13.18: eksempler/lydApplet2.html

```
<html>
  LydApplet2:
  <hr>
  <applet code="LydApplet2.class" width="300" height="100">
  </applet>
</html>
```

- http://matiksi.hive.no/oopva60/eksempler/lydApplet2.html

`http://matiksi.hive.no/oopva60/eksempler/lydApplet2.html`

- eksempler/LydApplet2.java

Listing 13.19: eksempler/LydApplet2.java

```
public class LydApplet2
extends javax.swing.JApplet
implements java.awt.event.ActionListener {

  javax.swing.JButton spill, stopp, loop;
  javax.swing.JPanel panel;
  java.applet.AudioClip lyd;


  public void init(){

    try {

      lyd = getAudioClip(
          new java.net.URL(
              getDocumentBase(),
              "Tuxedomoon.Jingle4.wav"
          )
      );

    } catch (java.net.MalformedURLException e) {

      e.printStackTrace();
    }

    spill = new javax.swing.JButton("Spill");
    stopp = new javax.swing.JButton("Stopp");
    loop  = new javax.swing.JButton("Loop");

    spill.addActionListener(this);
    stopp.addActionListener(this);
    loop. addActionListener(this);

    panel = new javax.swing.JPanel();

    panel.add(spill);
```

```
37      panel.add(stopp);
38      panel.add(loop);
39
40      this.add(panel);
41    }
42
43    public void actionPerformed(
44        java.awt.event.ActionEvent e) {
45
46      Object o = e.getSource();
47
48      if (o == spill)
49        lyd.play();
50
51      else if (o == stopp)
52        lyd.stop();
53
54      else if (o == loop)
55        lyd.loop();
56    }
57  }
```

13.6 Oppgave-eksempler

- Definer et interface for et fjernt objekt som ... (gjør et eller annet)

```
A a = (A) Naming.lookup("rmi://matiksi.hive.no/A");
B b = (B) inn.readObject(); // inn er av klassen 'ObjectInputStream'
</pre>

Både A og B har metoden getC. Beskriv kort hovedforskjellene mellom å kjøre <i>a.getC
```

- Beskriv linje for linje hva som skjer i når følgende kode blir kjørt: ... (kode kommer her) ...

- Hva blir utskriften av følgende programkode: ... (kode kommer her) ...

- Under ser du en liste over ulike deler av en applikasjon som du skal være med på å utvikle. Bestem for hver komponent om den bør implementeres som applet, servlet, JSP eller bønne (bean). Begrunn svaret. ... ...

Kapittel 14

Eksamen våren 2009

Forsøk å besvare oppgavene under kort og presist. Når du i dette oppgavesettet blir bedt om å beskrive hvordan du ville skrevet/endret kode, er det en god i idé å inkludere kode i svaret.

14.1 Oppgave 1 (25%)

Tenk deg at du skal være med på å utvikle en web-applikasjon. Under ser du en liste over ulike deler av en applikasjonen. Bestem for hver komponent om den bør implementeres som applet, servlet, JSP eller bønne (bean). Oppgi kortfattede begrunnelser for valgene.

14.1.1 a)

Komponent som oppdaterer vare-lager-databasen

14.1.2 Løsningsforslag

Dette er en komponent som ikke trenger noe UI. Det er dermed ikke noe poeng å bruke applet er eller JSP. En servlet/bønne kan dermed være passende.

14.1.3 b)

Komponent som utfører penge-transaksjoner

14.1.4 Løsningsforslag

Dette er en komponent som ikke trenger noe UI. Applet og JSP er dermed ikke aktuelt. En servlet/bønne kan dermed være passende.

14.1.5 c)

En komponent som skriver logg til fil

14.1.6 Løsningsforslag

Loggfiler lagres på server. Applet er dermed ikke aktuelt. Dette er en komponent som ikke trenger noe UI. Derfor velger jeg ikke JSP. En servlet/bønne kan dermed være passende.

14.1.7 d)

Brukergrensesnitt for å betale

14.1.8 Løsningsforslag

Dette er en UI-komponent som trenger gode sikkerhetstiltak på kommunikasjonen mellom klient og tjener. En applet kan dermed være passende.

14.1.9 e)

Brukergrensesnitt for å handle

14.1.10 Løsningsforslag

Her er det mye kommunikasjon mellom vare-database og bruker, men ikke så høye krav til sikkerhet som i d). Mye kommunikasjon mellom vare-databasen og komponeneten tilsier at den bør ligge på serveren, altså velger vi ikke applet. Siden det er mye kommunikasjon med bruker vil jeg bruke JSP for å lage grensesnittet.

14.2 Oppgave 2 (25%)

14.2.1 a)

Gi et eksempel på en applikasjon hvor det er nyttig å kjøre flere konkurrerende tråder. Oppgi en kortfattet begrunnelse.

14.2.2 Løsningsforslag

En nettleser viser ofte frem sider som består av flere komponenter samtidig. Noen av disse kan ta lengre tid å laste ned. Ved å kjøre nedlastingene i hver sin tråd, kan applikasjonen presentere ferdige komponenter etter hvert som de blir lastet ned. En tråd kan også lese brukerens handlinger, uten å måtte vente til alt innhold er lastet.

14.2.3 b)

Gi et eksempel på en applikasjon hvor det *ikke* er nyttig å kjøre flere konkurrerende tråder. Oppgi en kortfattet begrunnelse.

14.2.4 Løsningsforslag

En applikasjon som fører brukeren gjennom en prosedyre trinn for trinn, slik at det trinn 2 forutsetter at trinn 1 er fullført, o.s.v. Her vil ikke flere tråder gi gevinst, da alt må gjøres sekvensielt uansett.

14.2.5 c)

Gi et eksempel på en applikasjon som kjører flere konkurrerende tråder og har behov for synkronisering. Oppgi en kortfattet begrunnelse.

14.2.6 Løsningsforslag

En tjener som lar flere flere klienter skrive i en logg, har behov for å synkronsiere, slik at innslagene holdes fra hverandre og ikke blandes sammen.

14.2.7 d)

Gi et eksempel på en applikasjon som kjører flere konkurrerende tråder og *ikke* har behov for synkronisering. Oppgi en kortfattet begrunnelse.

14.2.8 Løsningsforslag

Eksemplet i a) har ikke behov for synkronisering, da poenget med trådene her er at de skal være uavhengige av hverandre.

14.2.9 e)

Beskriv kort, og vis med et kode-eksempel, hvordan synkronisering av tråder implementeres i java.

14.2.10 Løsningsforslag

Synkronisering av tråder kan utføres ved at objekt inneholder funksjoner som deklareres som *syncronized*. Trådene som skal synkroniseres kaller disse metodene ved dette objektet.

Eksempelet i c) kan ha et objekt av en klasse Logg

```
1  public class Logg{
2
3      public syncronized void skrivLogg(String){
4
5        // her kommer kode for å skrive til loggen
6      }
7  }
```

Klientene kan betjenes av tråder som kaller opp metoden skrivLogg() i dette objektet.

14.3 Oppgave 3 (25%)

```
1  public class Eksempel {
2
3      public static void main(String[] args) throws java.io.IOException {
4
5    java.net.ServerSocket ss = new java.net.ServerSocket(8888);
6    java.net.Socket       s;
7    java.io.PrintWriter   u;
8    java.util.Scanner     i;
9
10   while (true) {
11
12       s = ss.accept();
13       u = new java.io.PrintWriter(s.getOutputStream());
14       i = new java.util.Scanner(s.getInputStream());
15
```

```
16          while(true) {
17        u.format("%s\n", i.nextLine() );
18        u.flush();
19          }
20     }
21        }
22  }
```

14.3.1 a)

Gi en kort overordnet beskrivelse av hva koden gjør

14.3.2 Løsningsforslag

Når denne koden utføres vil det startes en tcp-server som lytter på port 8888. Når en klient har koblet seg til, vil tjeneren sende dataene tilbake, som et ekko.

14.3.3 b)

Gi en detaljert beskrivelse, linje for linje, av hva som skjer i når koden i linjene 10-20 kjøres.

14.3.4 Løsningsforslag

14.3.4.1 10

En ytre løkke, som gjentas helt til tjeneren avslutter, defineres.

14.3.4.2 12

Metoden accept() kjøres på serversocketen (ss) som er satt til å lytte på port 8888 i linje 5. Her vil tråden blokkers inntil en klient har forbundet seg til porten. En ny socket som for forbinder tjeneren med klienten opprettes da. En referanse til denne socket'n returneres fra accept() og lagres i variablen 's'.

14.3.4.3 13

Metoden getOutputStream() kalles fra socket'en forbundet med klienten. Dette returnerer en referanse til en OutputStream() som igjen brukes som argument til en PrintWriter-konstruktør. Returen fra denne konstruktøren gir en referanse til en instans av PrintWriter koblet til klient. Denne referansen kan nå brukes til å skrive til klienten.

14.3.4.4 14

Metoden getInputStream() kalles fra socket'en forbundet med klienten. Dette returnerer en referanse til en InputStream() som igjen brukes som argument til en Scanner-konstruktør. Returen fra denne konstruktøren gir en referanse til en instans av Scanner koblet til klient. Denne referansen kan nå brukes til å lese fra klienten.

14.3.4.5 16

En indre løkke, som gjentas helt til klienten har avsluttet kommunikasjonen, defineres

14.3.4.6 17

Referansen til Scanner-instansen brukes til å lese en linje fra klienten v.h.a. metoden *nextLine()*. Linjen skrives tilbake til klienten v.h.a. Scanner-metoden *format()*.

14.3.4.7 18

For å sikre at innholdet blir sendt, blir socket'ens utbuffer tømmes.

14.3.4.8 19

Den indre løkka avsluttes.

14.3.4.9 20

Den ytre løkka avsluttes.

14.3.5 c)

Hva blir utskriften av koden?

14.3.6 Løsningsforslag

Det klienten skriver til tjeneren blir skrevet tilbake til klienten. Ingenting blir skrevet ut lokalt hos tjeneren.

14.3.7 d)

Hvordan ville du endret koden slik at den ble *ikke-blokkerende* (non-blocking). Vær konkret - skriv gjerne kode.

14.3.8 Løsningsforslag

For å opprette en ikke-blokkerenede socket brukes java.nio.channels.Serversocket:

```
1 sSC = ServerSocketChannel.open();
2 sSC.configureBlocking(false);
3 ServerSocket sS = sSC.socket();
4 sS.bind( InetSocketAddress(8888) );
```

Vi bruker en *selector*, hvor vi for hver socket registrerer hvilke operajsoner vi vil overvåke.

```
1   sel = java.nio.channels.Selector.open();
2   sSC.register(sel, SelectionKey.OP_ACCEPT);
```

I en evig løkke gjøres det blokkerende kallet *sel.select()* som returnerer når noen av de registrerte operasjonene er har returnert noe. De ulike hendelsene håndeteres v.h.a av et intererbart *nøkkelsett*, (*hendelsesNokler = sel.selectedKeys()*). Nøklene må fjernes etter bruk.(*sel.selectedKeys().remove(nokkel)*);

Accept() vil returnere en ny socket, som også må settes opp som ikke-blokkerende. For denne må lese-operasjonen registreres i selectoren.

14.3.9 e)

Hvordan ville du endret koden slik at den tok i bruk *Remote Invocation Method*. Vær konkret - skriv gjerne kode. Pass på å få med beskrivelse av nødvendige komponenter.

14.3.10 Løsningsforslag

```
// Grensensittet som skal implementeres og gjoeres kjent for klientene
public interface Ekkogrensesnitt extends java.rmi.Remote{

  public String ekko(String tekst) throws java.rmi.RemoteException;
}


// Implementasjon av grensesnittet
public class EkkoImpl
extends java.rmi.server.UnicastRemoteObject
implements Ekkogrensesnitt{

  public EkkoImpl() throws java.rmi.RemoteException {

  }

  public String ekko(String tekst) throws java.rmi.RemoteException {

    return tekst;
  }
}

// Selve tjeneren starter navneregisteret og instansierer implementasjonen.
// Objektet (sammen med et navn) registreres i navneregisteret.
public class EkkoTjener {

  public static void main (String[] args) throws
  java.net.MalformedURLException,
  java.rmi.RemoteException{

    java.rmi.registry.LocateRegistry.createRegistry(1099);
    java.rmi.Naming.rebind(
        "rmi://vert.domene/Ekko",
        new EkkoImpl()
    );
  }
}
```

14.4 Oppgave 4 (25%)

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html><head><title>Eksempel</title></head>
  <body>
    <form action=eksempel>
      <input type=text name=A>
      <input type=text name=B>
      <input type=text name=C>
      <input type=submit>
    </form>
```

```
10    </body>
11 </html>
```

14.4.1 a)

Gi en kort overordnet beskrivelse av koden

Dette er et HTML-dokument som inneholder kode som instruerer en nettleser til å tegne opp et skjema med tre tekstfelt (A,B,C) og en trykk-kapp for å sende det innfylte skjema tilbake til web-serveren.

14.4.2 b)

Hvordan ville du skrevet et CGI-program i java som behandlet skjemaet i koden. Vær konkret - skriv gjerne kode. Pass på å få med beskrivelse av nødvendige komponenter.

14.4.3 Løsningsforslag

Skjemaet er tilgjengelig for CGI-programmene i miljøvariablen QUERY_STRING. I java kan vi referere til denne med *System.getenv("QUERY_STRING")*. Denne tekst-strengen er URL-kodet. For å dekode kan metoden *java.net.URLDecoder.decode()*. F.eks. slik:

```
1   String queryString   = System.getenv("QUERY\_STRING")
2   String decodedString = java.net.URLDecoder.decode(queryString,"UTF-8")}
```

emphdecodeString vil da inneholde ent liste med variablene som navn-verdi-par. Hvis teksteltet A inneholdt verdien "Per", B inneholdt Pål" og C innholdt "Espen", ville innholdet i queryStringen vært slik:

A=Per&B=Pål&C=Espen

Tekststrengen kan nå videre behandles f.eks. ved at det splittes opp og lagres i ulike variabler.

14.4.4 c)

Hvordan ville du skrevet en servlet som behandlet skjemaet i koden. Vær konkret - skriv gjerne kode.

14.4.5 Løsningsforslag

```
1  public class Serveline  extends javax.servlet.http.HttpServlet {
2
3      public void doGet(
4          javax.servlet.http.HttpServletRequest  request,
5          javax.servlet.http.HttpServletResponse respons )
6
7          throws java.io.IOException,
8                 javax.servlet.ServletException {
9
10         String A = request.getParameter("A");
11         String B = request.getParameter("B");
12         String C = request.getParameter("C");
13     }
14 }
```

14.4.6 d)

Oppgi (kortfattet) fordeler ved å implementere skjema-behandlingen som et CGI-program.

14.4.7 Løsningsforslag

Alle de mest brukte web-serverne har støtte for CGI, dermed kan et CGI-program brukes på alle web-servere sålenge det er mulig å kjøre en java-virtuell-maskin på dem.

14.4.8 e)

Oppgi (kortfattet) fordeler ved å implementere skjema-behandlingen som en servlet.

14.4.9 Løsningsforslag

Enklere å skrive/lese kode. Slipper overhead ved fork() for hver klient-forespørsel