# Notater til OOPVA60, andre utgave

Thomas Nordli <tn@hive.no>

24. mai 2011

Kompendium 1 / 2011 Høgskolen i Vestfold

Issn 0808-1328

# Innhold

# Om skriftet

Skriftet *Notater til OOPVA60, andre utgave* er et kompendium som inneholder undervisningsmateriale. Undervisningsmaterialet er produsert i forbindelse med undervisning i faget *Objektorientert programmering II.* , våren 2011. Faget ble tilbudt som valgfag, for

studenter i *Oslofjordalliansen* med *grunnleggende ferdigheter objektorientert programmering.*

Kompendiet er en sammenstilling av:

- notater
- eksempler
- oppgaver
- løsningsforslag

, som ble brukt i forbindelse med undervisningen.

## Endringer i siden forrige utgave

Foruten små forbedringer her og der, inneholder denne utgaven flere løsningsforslag på øvingsoppgaver.

Pensum i kurset er endret slik at noen temaer er byttet ut med andre.

Følgende temaer er nye:

- Android
- JavaScript
- CSS

Følgende temaer er utgått fra kursets pensum.

- CORBA
- Applets

Applets er allikevel med som vedlegg.

## Om forfatteren

Forfatteren, Thomas Nordli, har vært fagansvarlig for kurset siden 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*. I tillegg er han lærer i NKI-kurset *Operativsystemet Linux*.

# 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"-notasjon: Fire 8-bits desimaltall adskilt med punktum.
- eks.: 128.39.112.18

#### IPv6-adresse: 128 bit

eks: 2001:700:2300:1e:250:56ff:feb4:47ed

- 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/01/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 aa fremskaffe ip-adressen til vert basert
10    * paa vertens domenenavn.
11    *
12    * Baade 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 aa 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 paa en annen maskin:"
33         );
34
35     String maskin = inngang.next();
36
37     /*
38      * For aa gjoere 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 gjoere et navneoppslag.
43      * (i DNS eller filen hosts).
44      *
45      *  Dersom 'vert' inneholder en tekststreng med ip-adresse,
46      *  gjoeres kun en sjekk paa om det er paa 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.

**For å teste programmet:**

- Start programmet med kommandoen 'java TCPSensor' i et konsoll-vindu.
- Dersom du prøver dette på en oopva60.hive.no (alias debbie) kan port 8888 være opptatt. Du må da endre kode slik at den benytter en ledig port. Portnummeret må være høyere enn 1024.
- Koble deg til TCP-port 8888 fra en annet program (f.eks. 'netcat' fra et annet konsoll-vindu).

Listing 1.2: eksempler/01/TCPSensor.java

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

### 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'.

- Du må skifte til en ledig port, hvis 8888 er opptatt.

Listing 1.3: eksempler/01/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

Listing 1.4: eksempler/01/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
23          utskriver.format(
24                "Interessant at du sier \"%s\".\n",
25                innleser.nextLine() );
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 i java

**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).

Listing 1.5: eksempler/01/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 paa port\t" +  datagramSocket.getLocalPort() );
15
16      datagramSocket.receive(datagram);
17
18      System.out.println(
19          "Datagrammet fra porten "    + datagram.getPort() +
20          " paa 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

- Skriv et "halloverden"-program i java. Kall filen 'Hallo.java'.
- Logg inn på oopva60.hive.no (alias debbie). Ta kontakt med faglærer hvis du ikke har konto eller kommer inn.
- Kjør "halloverden"-programmet fra kommandolinjen på debbie.

### 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())"

Listing 2.1: losninger/01/TCPSkravler_1v2.java

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

**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.

Listing 2.2: losninger/01/TCPKlient.java

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

## Oppgave 1.3

**Programmer en klient til eksempelprogrammet UDPSensor**

Listing 2.3: losninger/01/UDPKlient.java

```
1  public class UDPKlient {
2
3    public static void main(String[] args) throws java.io.IOException {
4
5      java.net.DatagramSocket s;
6      java.net.DatagramPacket d;
7
8      s = new java.net.DatagramSocket();
9      d = new java.net.DatagramPacket(
10         "test".getBytes(),
11         4,
```

```
12          java.net.InetAddress.getLocalHost(),
13          8888
14          );
15      s.send(d);
16      s.close();
17      }
18  }
```

## 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

Listing 2.4: eksempler/02/Traad_1.java

```
1  public class Traad_1 extends Thread {
2
3    public static void main(String[] args) throws
4    java.io.IOException {
5
6      Traad_1 t1, t2;
7
8      t1 = new Traad_1();
9      t2 = new Traad_1();
10
11     t1.start();
12     t2.start();
13   }
14
15   public void run(){
```

```
16
17      int i;
18      for (i=0;i<5;i++)
19        System.out.println(this.getName());
20    }
21  }
```

Listing 2.5: eksempler/02/Traad_2.java

```
1  public class Traad_2 extends Thread {
2    public static void main(String[] args) throws
3    java.io.IOException {
4
5      int i;
6      for (i=0; i<5; i++)
7        new Traad_2().start();
8    }
9
10   public void run(){
11
12     int i;
13     for (i=0;i<5;i++) {
14       System.out.println(this.getName());
15       try {
16         sleep( (int)(Math.random()*1000) );
17
18       } catch (InterruptedException e) {
19         e.printStackTrace();
20       }
21     }
22   }
23 }
```

**implements Runnable**

Listing 2.6: eksempler/02/Traad_3.java

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

```
23            e.printStackTrace();
24          }
25        }
26      }
27    }
```

## 2.3 RaceCondition

### Eksempler

Listing 2.7: eksempler/02/Balanse_1.java

```
1  public class Balanse_1 extends Thread{
2
3    static final int T = 1000; // Totalsummen i spillet
4    static final int N = 5;    // Antall konti
5    static private int konto[] = new int[N];
6
7    java.util.Random slumptall = new java.util.Random();
8
9    public static void main(String[] args)  {
10
11     opprettKonti();
12     new Balanse_1().start();
13
14     while(true){
15
16       try {
17         sleep(1000);
18       } catch (InterruptedException e) {
19         e.printStackTrace();
20       }
21
22       skriv_saldo();
23     }
24   }
25
26   public void run()  {
27
28     int til, fra, belop;
29
30     while(true){
31
32       fra=Math.abs( slumptall.nextInt() ) %N;
33       til=Math.abs( slumptall.nextInt() ) %N;
34       belop=Math.abs( slumptall.nextInt() ) % ( (konto[fra]/10) );
35
36       konto[fra]-=belop;
37       konto[til]+=belop;
38     }
39   }
40
41   private static void opprettKonti(){
42
43     int i;
44     for (i=0;i<N;i++)
45       konto[i]=T/N;
46
```

```
47      skriv_saldo();
48    }
49
50    private static void skriv_saldo(){
51
52      int sum,i;
53
54      for (i=0, sum=0; i<N; i++){
55
56        System.out.printf("konto %d:\t%d\n", i, konto[i]);
57        sum+=konto[i];
58      }
59
60      System.out.printf("--------------\nTotalt:\t%d\n==============\n",sum);
61    }
62  }
```

Listing 2.8: eksempler/02/Balanse_2.java

```
1   public class Balanse_2 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
12    InterruptedException  {
13
14      opprettKonti();
15      opprettSpekulanter();
16
17      while (true){
18        skriv_saldo();
19        sleep(1000);
20      }
21    }
22
23    private static void opprettSpekulanter(){
24
25       int i;
26       for (i=0; i<S; i++)
27         new Balanse_2().start();
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    private static void opprettKonti() throws
46    InterruptedException{
47
48      int i;
49
50      for (i=0;i<N;i++)
51        konto[i]=T/N;
52
53      skriv_saldo();
54    }
55
56    private static void skriv_saldo() throws
57    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    }
68  }
```

Listing 2.9: eksempler/02/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
12    InterruptedException  {
13
14      opprettKonti();
15      opprettSpekulanter();
16
17      while (true){
18        skriv_saldo();
19        sleep(1000);
20      }
21    }
22
23    private static void opprettSpekulanter(){
24
25       int i;
26       for (i=0; i<S; i++)
27         new Balanse_3().start();
28    }
```

```
29
30    public void run()  {
31
32      while(true)
33        flyttPenger();
34    }
35
36    private static synchronized void flyttPenger() {
37
38      int til, fra, belop;
39
40      fra=Math.abs( slumptall.nextInt() ) %N;
41      til=Math.abs( slumptall.nextInt() ) %N;
42      belop=Math.abs( slumptall.nextInt() ) % ( (konto[fra]/10) );
43
44      konto[fra]-=belop;
45      konto[til]+=belop;
46    }
47
48    private static synchronized void opprettKonti() throws
49    InterruptedException{
50
51      int i;
52
53      for (i=0;i<N;i++)
54        konto[i]=T/N;
55
56      skriv_saldo();
57    }
58
59    private static synchronized void skriv_saldo() throws
60    InterruptedException{
61
62      int sum, i;
63
64      for (i=0, sum=0; i<N; i++){
65        System.out.printf("konto_%d:\t%d\n", i, konto[i]);
66        sum+=konto[i];
67      }
68
69      System.out.printf("---------------\nTotalt:\t%d\n===============\n",sum);
70      System.out.flush();
71    }
72  }
```

## 2.4 Øvelsesoppgaver

### 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!

Listing 2.10: eksempler/02/TCPSkravler_3.java

```
1 public class TCPSkravler_3 extends Thread{
```

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

## 2.2

Skriv om eksemplet 'Balanse_3', slik at det består av to klasser: Bank og Spekulant.

# Kapittel 3

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

## 3.1 Løsningsforslag på forrige øvelsesoppgave

### 2.1

- 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.

Listing 3.1: losninger/02/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("
                       debbie.hive.no"),8888);
10                 java.io.PrintWriter ut  = new java.io.PrintWriter(s.
                       getOutputStream());
11                 java.util.Scanner inn = new java.util.Scanner(System.in);
12
13                 while (inn.hasNextLine()){
14                         ut.println(inn.nextLine());
15                         ut.flush();
16                 }
17
18                 new Lytter(s).start();
19         }
20 }
```

Listing 3.2: losninger/02/Lytter.java

```
1  public class Lytter extends Thread{
2
3    private java.util.Scanner inn;
```

```
4
5     Lytter(java.net.Socket t) throws java.io.IOException {
6
7       inn = new java.util.Scanner(t.getInputStream());
8     }
9
10    public void run() {
11
12        while (inn.hasNextLine())
13          System.out.println(inn.nextLine());
14    }
15  }
```

## 2.2

- Skriv om eksemplet 'Balanse_3', slik at det består av to klasser: Bank og Spekulant.

Listing 3.3: losninger/02/Bank.java

```
1  public class Bank {
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    public static void main(String[] args) throws
10   InterruptedException  {
11
12     java.util.Scanner stdin = new java.util.Scanner(System.in);
13
14     // Fordeler totalsummen på kontiene
15     for (int i=0;i<N;i++)
16       konto[i]=T/N;
17
18     // Starter spekulantene
19     for (int i=0; i<S; i++)
20        new Spekulant().start();
21
22     // Skriver saldo for hvert linjeskift fra STDIN
23     while (stdin.hasNextLine()){
24       stdin.nextLine();
25       skriv_saldo();
26     }
27   }
28
29   public static synchronized void skriv_saldo() throws
30   InterruptedException{
31
32     int sum, i;
33
34     for (i=0, sum=0; i<N; i++){
35       System.out.printf("konto %d:\t%d\n", i, konto[i]);
36       sum+=konto[i];
37     }
38
```

```
39      System.out.printf(
40          "----------\nTotalt:\t%d\n==========\n",
41          sum
42          );
43    }
44
45    public static synchronized void flyttPenger(int belop, int fra, int til)
          {
46
47      konto[fra]-=belop;
48      konto[til]+=belop;
49    }
50  }
```

Listing 3.4: losninger/02/Spekulant.java

```
1
2   public class Spekulant extends Thread{
3
4     static java.util.Random slumptall = new java.util.Random();
5
6     public void run()  {
7
8       while(true){
9         int til, fra, belop;
10
11        fra=Math.abs( slumptall.nextInt() ) %Bank.N;
12        til=Math.abs( slumptall.nextInt() ) %Bank.N;
13        belop=Math.abs( slumptall.nextInt() ) % ( (Bank.konto[fra]/10) );
14
15        Bank.flyttPenger(belop, fra, til);
16      }
17    }
18  }
```

## 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

Listing 3.5: eksempler/03/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  }
```

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.

Listing 3.6: eksempler/03/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   }
```

## 3.3 URL

- Uniform Resource Locator
- Definerer hvor på internett og hvordan
- <protokoll>://<vertsnavn>[:<port>]/absolutt_sti[?argumenter]
- Eksemplet WebLaster.java demonsterer bruk av URL. For å lagre websiden på en fil kan følgende kommando kjøres fra kommandolinja: java WebLaster > filnavn.html.

Listing 3.7: eksempler/03/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 }
```

## 3.4 Lesing av fil

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

Listing 3.8: eksempler/03/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 }
```

## 3.5 Kommandolinje-argumenter

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

Listing 3.9: eksempler/03/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 }
```

## 3.6 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.

Listing 4.1: losninger/03/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.

Listing 4.2: losninger/03/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 }
```

Listing 4.3: losninger/03/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.

Listing 4.4: losninger/03/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  }
```

Listing 4.5: losninger/03/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.

Listing 4.6: eksempler/04/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  }
```

Listing 4.7: eksempler/04/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 }
```

Listing 4.8: eksempler/04/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.

Listing 4.9: eksempler/04/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 }
```

Listing 4.10: eksempler/04/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.

Listing 4.11: eksempler/04/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 }
```

Listing 4.12: eksempler/04/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).

Listing 5.1: losninger/04/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.

Listing 5.2: losninger/04/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 }
```

Listing 5.3: losninger/04/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 Grunnleggende HTML

### HTML

- HyperText Markup Language
- ”tagge-språk” for web-dokumenter
- Utviklet på CERN (Berners-Lee og Connolly)
- Standarisert av IETF (Internet Engineering Task Force) i 1995 (RFC 1866)

**Utviklet med tanke på plattformuavhengighet**

- PC'er
- telefoner
- PDA'er
- etc. ...

## HTML 5

- HTML 5 - i ferd med å implementeres
- HTML 5 skal erstatte HTML 4.01
- Inneholder blant annet egne tagger for lyd og video
- Eksemplet viser oppbygningen av et html-dokument med hode og kropp.

Listing 5.4: eksempler/05/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 HTTP

`http://www.ietf.org/rfc/rfc1945`

- HyperText Transfer Protocol
- Protokoll for filoverføring mellom nettlesere og web-tjenere
- basert på "request-respose"-paradigme

### Stadier i en HTTP-transaksjon

- 1. Forbindelse (connection) - klient kobler seg til tjeneren med TCP (vanligvis port 80).
- 2. Forespørsel (request) - klient spør tjener
- 3. Respons - tjener svarer klient
- 4. Steng (close) - tjener stenger forbindelsen

### tilstandsløs

- Tjeneren lagrer ingen informasjon om klientenes forespørsler
- Vanskelig å støtte konseptet 'sesjon' som er essensielt for basis-DBMS-transaksjoner

### HTTP-forespørsel

#### Hode (header)

- forespørsel-type (f.eks GET, POST, PUT HEAD)
- navn/sti på "ressurs" (f.eks /losninger/03/logg)
- HTTP-versjonen (fra 1995 HTTP/1.0 – før det: HTTP/0.9)

#### eventuelt kropp

adskilt fra hodet med en tom linje

### HTTP-respons

#### Hode

- HTTP-versjon
- Status
- Info for kontroll av klientens oppførsel (f.eks. mimetype)

**eventuelt kropp**

adskilt fra hodet med en tom linje

### Demonstrasjon av http-transaksjon

```
echo -e "GET http://oopva60.hive.no/ HTTP/1.0\n\n" | nc oopva60.hive.no 80
```

## 5.4 Grunnleggende CGI

- Common Gateway Interface
- Spesifikasjon som definerer hvordan web-tjener og CGI-skript/programmer kommuniserer.
- URL refererer til en fil som kjøres av web-tjener. Utskriften av programmet sendes i HTTP-responsen.

### Web-tjener vedlikeholder en del miljøvariabler som skriptet arver

- CGI-program som lister ut alle miljø-variablene det har arvet.

Listing 5.5: eksempler/05/env.cgi

```
1 #!/bin/sh
2 java Env
```

Listing 5.6: eksempler/05/Env.java

```
1  public class Env {
2
3      public static void main(String[] args) {
4
5    System.out.printf("Content-type:text/plain\n\n");
6
7    for (java.util.Map.Entry<String, String> e : System.getenv().entrySet() )
8        System.out.println(e.toString());
9      }
10 }
```

### Hallo_1

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

Listing 5.7: eksempler/05/hallo_1.html

```
1 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
2
3 <html><head><title>Hallo_1</title></head>
4   <body>
5
6     Si hallo til:
```

```
7       <ul>
8
9         <li>
10          <a href="hallo_1.cgi?Arne">Arne</a>
11        <li>
12          <a href="hallo_1.cgi?Bjarne">Bjarne</a>
13        <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>
```

Listing 5.8: eksempler/05/hallo_1.cgi

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

Listing 5.9: eksempler/05/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.

Listing 5.10: eksempler/05/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>
```

Listing 5.11: eksempler/05/hallo_2.cgi

```
#!/bin/sh

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

Listing 5.12: eksempler/05/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.

Listing 5.13: eksempler/05/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>
```

Listing 5.14: eksempler/05/hallo_3.cgi

```
#!/bin/sh

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

Listing 5.15: eksempler/05/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(
```

```
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.

Listing 5.16: eksempler/05/hallo_4.cgi

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

Listing 5.17: eksempler/05/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  }
```

## 5.5 HTML for android plattformen

### To måter å levere applikasjoner til android-plattformen

- 1. Klientside-applikasjon
- 2. Web-applikasjon
- Fra og med Android 2.0, finnes det i standardnettleser-app'en og 'view widget'en WebView, noen muligheter for å "android-tilpasse" web-applikasjoner.
- Android kategoriserer skjermer basert på punkt-tetthet.
- Tetthet(density) måles i antall punkter pr. tomme (dpi - dot per inch).

### Tre kategorier av tetthet støttes i plattformen

- lav (ldpi)
- medium (mdpi)
- høy (hdpi)

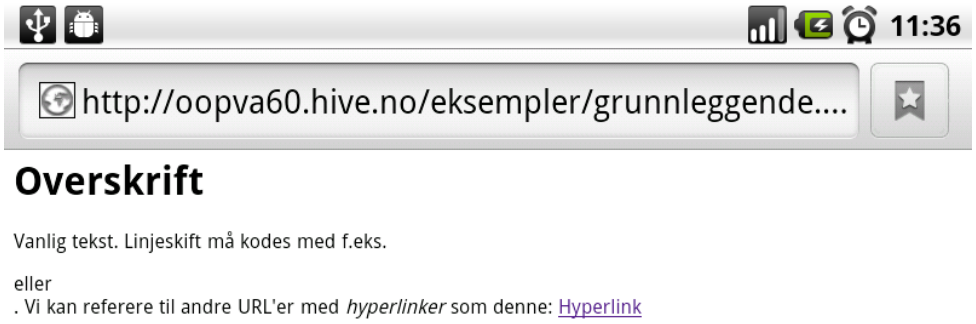
### viewport

- Området hvor web-sider vises (i standardnettleseren og WebView) kalles 'viewport'.
- Det er definert egenskap for html-taggen 'meta'. Denne egenskapen gir oss muligheter til å justere størrelsen på websiden basert på terminalens skjermstørrelse.

```
<meta name="viewport"
content="
            height = [pixel_value | device-height] ,
            width = [pixel_value | device-width ] ,
            initial-scale = float_value ,
            minimum-scale = float_value ,
            maximum-scale = float_value ,
            user-scalable = [yes | no] ,
            target-densitydpi = [dpi_value | device-dpi |
                                 high-dpi | medium-dpi | low-dpi]
">
```

## Eksempel

### eksempler/grunnleggende.html




Listing 5.18: eksempler/05/mobil.html

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE
    html PUBLIC "-//W3C//DTD XHTML Basic 1.1//EN"
    "http://www.w3.org/TR/xhtml-basic/xhtml-basic11.dtd"
    >

<html>

  <head>

    <title> Tittel </title>
    <meta
        name='viewport'
        content='width=device-width, user-scalable=yes, initial-scale=1.2'
        />
  </head>

  <body>

    <h1> Overskrift </h1>
    <p>
      Vanlig tekst.
      Linjeskift maa kodes med f.eks. &lt;p&gt; eller &lt;br&gt; <br/>.
      Vi kan referere til andre URL'er med <i>hyperlinker</i> som denne:
      <a href="http://www.hive.no">
  Hyperlink
      </a>
    </p>
  </body>

</html>
```

**eksempler/mobil.html**


## 5.6 Øvelser

### 5.1

- Bli kjent med systemene.

**a)**

- Prøv å få et "halloverden" cgi-skript til å kjøre på ditt område på debbie.hive.no.
- Skriptet som starter den virtuelle maskinen, må ha fil-etternavn '.cgi'.
- Skriptet må ligge i underkatalogen 'public_html' i hjemmekatalog.
- Skriptet må være kjørbart for alle med konto på
- Hvis programmet til arne heter "/home/arne/public_html/prg.cgi", vil URL'n til programmet være: http://debbie.hive.no/ arne/prg.cgi
- Web-området er beskyttet. Navn og passord er det samme du brukte for å komme til disse notatene.

**b)**

- Prøv eksemplene 'eksempler/grunnleggende.html' og 'eksempler/mobil.html' i standardnettleseren på en android-eumulator.
- På lab'en kan androidemulatoren startes fra eclipse (Window -> Start Android SDK and AVD Manager)
- Hvis du ikke er på lab'en, kan du installere "Android SDK and AVD Manager" ved å følge instruksjonene du finner ved å følge lenken under:

`http://developer.android.com/sdk/installing.html`

## 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.

## 5.3 (Obligatorisk)

### a)

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

### b)

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

### c)

- Utvid en (eller begge) programmene du lagde i a) og b) slik de vises ulikt på pc og mobiltelefon. Sørg for at fremvisning på android-enhet, tilbasses skjermstørrelsen.
- Sørg for at mobil-fremvisngen ikke bryter retningslinjene du finner ved å følge lenken under:

`http://developer.android.com/guide/webapps/best-practices.html`

# Kapittel 6

# Android I

## 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.

Listing 6.1: losninger/05/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 Android-pensum

- Android-relatrert pensum er hentet fra The Developer's Guide (på developer.android.com).

### Til forrige forelesning

**Using Viewport Metadata (gjelder stoff fra forrige forelesning som bl.a. trengs til oblig )**

`http://developer.android.com/guide/webapps/targeting.html#Metadata`

#### Best Practices for Web Apps

`http://developer.android.com/guide/webapps/best-practices.html`

### Til denne forelesningen

#### What Is Android?

`http://developer.android.com/guide/basics/what-is-android.html`

#### Application Components

`http://developer.android.com/guide/topics/fundamentals.html#appcomp`

#### Developing In Eclipse, with ADT - unntatt 'Working with Library Projects'

`http://developer.android.com/guide/developing/eclipse-adt.html`

#### Developing In Other IDEs - unntatt 'Working with Library Projects'

`http://developer.android.com/guide/developing/other-ide.html`

#### Web Apps Overview

`http://developer.android.com/guide/webapps/index.html`

#### Building Web Apps in WebView

`http://developer.android.com/guide/webapps/webview.html`

## 6.3 Hva er Android?

- Komplett programvare-stack for mobile enheter.
- OS, mellomvare og nøkkelapplikasjoner
- Basert på åpen kildekode

### Eid av

`http://www.openhandsetalliance.com/`

- open handset alliance - konsortium av mange selskaper. 79 selskaper i skrivende stud.

### Arkitektur

#### figur

`http://developer.android.com/images/system-architecture.jpg`

### Applikasjoner

- Diverse applikasjoner følger med
- Hvis man oppretter konto hos Google, får man tilgang til Android Marked, hvor det finnes et vell av "apps".
- Man kan laste ned .apk-filer fra andre steder. Det finnes f.eks. alternative "markeder".
- Man kan lage sine egne applikasjoner

### Applikasjonsrammeverk

- "Views" - GUI-elementer som tekstfelt, trykk-knapper, etc.
- "Content Providers" - tilbyr datatilgang til andre applikasjoner.
- "Resource Manager" - tilgang til resursser som tekststrenger, bilder, layout-filer. etc.
- "Notification Manager" - styring av alarmer i statusfeltet.
- "Activity Manager" - håndterer applikasjonenes livssyklus og tilbyr felles bakoverstakk ("backstack") for navigasjon i aktivitets-historien.

### Programvare-bibliotek og "Android Runtime"

### Biblikotek

- "System C library" - Standard C system bibliotek tilpasset integrerte ("embedded") linuxbaserte enheter.
- SSL

**SQLite** `http://sqlite.org/`

- relasjonsdatabase-motor

**WebKit** `http://webkit.org/`

- nettlesermotor (web browser engine)
- brukes også av Nettleserne Safari og Google Chrome.

**OpenGl** `http://www.opengl.org/`

- Open Graphics Library
- Kryssplattform API-standard for 2D- og 3D-grafikk.

**etc.** ...

### Android Runtime

**Dalvik VM** `http://en.wikipedia.org/wiki/Dalvik_%28software%29`

- virtuell maskin som kjører filer i formatet Dalvik Executeable (.dex)
- verktøyet dx kompilerer .dex-filer av .class-filer.
- Hver applikasjon kjører en instans av Dalvik i en egen prosess.

**Kjerne bibliotek** `http://developer.android.com/reference/packages.html`

- tilbyr det meste av av kjernebibliotekene i programmeringsspråket Java.

**Operativsystem-kjerne (linux)**

`http://kernel.org/`

- Linux versjon 2.6
- Abstraksjonslag mellom maskinvaren og øvrig programvare.
- Systemtjenester som sikkerhet, minnehåndtering, prosesshåndtering, nettverks-stakk og modell for enhetsdrivere (device drivers).

## 6.4 Applikasjons-komponenter

### Activity

`http://developer.android.com/reference/android/app/Activity.html`

### Services

`http://developer.android.com/reference/android/app/Service.html`

### Broadcast receivers

`http://developer.android.com/reference/android/content/BroadcastReceiver.html`

### Content providers

`http://developer.android.com/reference/android/content/ContentProvider.html`

## 6.5 Installere utviklingsmiljø

### SDK Tools

- Under finner du instruksjonene jeg brukte for å installer 'sdk'et på debbie:
- wget http://dl.google.com/android/android-sdk_r08-linux_86.tgz
- tar xzvf android-sdk_r08-linux_86.tgz

- sudo mv android-sdk-linux_86/ /usr/local/
- cd /usr/local
- sudo ln -s android-sdk-linux_86/ android-sdk
- La til i '/etc/profile': PATH=${PATH}:"/usr/local/android-sdk/tools"
- sudo chown -R root.staff /usr/local/android-sdk-linux_86/

### Plattformer og "Samples"

- Under finner du instruksjonene jeg brukte for å installer 'plattformer' og 'samples' et på debbie:
- sudo /usr/local/android-sdk/tools/android
- GUI-vindu popper opp. Jeg velger alle komponentene fra Android-repositoriet, og installerer dem.

### Eclipse-plugin

- For oppskrift på installasjon av eclipse-plugin, følg lenken under.

`http://developer.android.com/sdk/eclipse-adt.html#installing`

## 6.6 Opprette AVD (Android Virtual Device)

- android create avd -t 4 -n VirtAndroid2.2

```
tn@debbie:~$ android create avd -t 4 -n minAVD -f -c 64M
Android 2.2 is a basic Android platform.
Do you wish to create a custom hardware profile [no]
Created AVD 'minAVD' based on Android 2.2,
with the following hardware config:
hw.lcd.density=160
tn@debbie:~$
```

- eller (fra android-GUI)

### Starte emulator

- emulator -avd minAVD
- eller (fra android-GUI)

## 6.7 Eksempel app's

### Hallo verden

- I dette eksemplet skal vi opprette lage en "hallo verden"-applikasjon fra kommandolinjen.

- For informasjon om hvordan dette gjøres fra Eclipse, se følgende referanse:

`http://developer.android.com/guide/developing/eclipse-adt.html`

#### finne TARGET_ID

android list targets

#### opprette prosjekt

- android create project

```
#!/bin/sh

T=4                 # Target_ID (Found with 'android list targets')
P=hallo             # Project directory set to working directory
A=HalloVerden       # Name of Activity
K=ofa.hallo # Package namspace

android create project -t $T -p $P -a $A -k $K
```

#### bygge

- ant debug

- hvis du skal publisere applikasjonen, og ikke bare kjøre applikasjonen i avlusningsmodus (debug-mode), må du signere den

#### installere fra debbie

lag en katalog kalt 'apks' i din hjemmekatalog og kopier apk-fila dit. Derfra kan du hente og installere den fra nettleseren på en android-enhet (eller emulator).

#### installere fra PC på usb-tilkoblet android-enhet

```
adb -d install bin/*-debug.apk
```

### Notat 1

- Dette eksemplet bruker en 'Intent' til å vise en webside.

Listing 6.2: eksempler/notat01/src/ofa/notat01/Notat01.java

```
1  package ofa.notat01;
2
3  public class Notat01 extends android.app.Activity
4  {
5      @Override
6      public void onCreate(android.os.Bundle savedInstanceState)
7      {
8    super.onCreate(savedInstanceState);
9
10   startActivity(
11           new android.content.Intent(
12               android.content.Intent.ACTION_VIEW,
13               android.net.Uri.parse("http://oopva60.hive.no/04.html")
14               )
15           );
16     }
17 }
```

## 6.8 WebView

`http://developer.android.com/reference/android/webkit/WebView.html`

- En utvidelse av klassen View
- Viser en webside eller webapplikasjon som en del av androidapplikasjonen.
- Ikke en fullverdig nettleser med navigasjonsknapper etc.

### Eksempler

#### Hallo web

- I eksemplet brukes WebView til å vise en tekststreng som en webside.

Listing 6.3: eksempler/halloweb/src/ofa/halloweb/HalloWeb.java

```
1  package ofa.halloweb;
2
3  public class HalloWeb extends android.app.Activity {
4
5      @Override
6    public void onCreate(android.os.Bundle savedInstanceState) {
7          super.onCreate(savedInstanceState);
8          android.webkit.WebView view = new android.webkit.WebView(this);
9          setContentView(view);
10         view.loadData("<h1>hallo <i>verden</i><h1>", "text/html", "utf-8");
11     }
12 }
```

### Notat 2

- Dette eksemplet viser bruk av WebView for å vise en webapplikasjon som ligger lagret lokalt på android-enheten i applikasjonens 'assets' katalog.

- Siden applikasjonen er lagret lokalt er den ikke avhengig av internettforbindelse.

Listing 6.4: eksempler/notat02/src/ofa/notat02/Notat02.java

```
1  package ofa.notat02;
2
3  public class Notat02 extends android.app.Activity
4  {
5      /** Called when the activity is first created. */
6      @Override
7      public void onCreate(android.os.Bundle savedInstanceState)
8      {
9    super.onCreate(savedInstanceState);
10
11   android.webkit.WebView wv = new android.webkit.WebView(this);
12   setContentView(wv);
13
14   wv.getSettings().setJavaScriptEnabled(true);
15   wv.getSettings().setBuiltInZoomControls(true);
16   wv.loadUrl("file:///android_asset/04.html");
17     }
18 }
```

## 6.9 Øvelser

### 6.1

- a) Lag en hallo-verden app fra kommandolinjen.

- b) Lag en hallo-verden-app fra Eclipse

### 6.2

#### a)

Lag en html-fil som inneholder noen hyperlenker til sider på weben.

#### b)

- Lag en android-applikasjon som bruker WebView til å vise frem html-fila fra a) og sidene den refererer til. Fila skal legges i assets-katalogen i prosjektkatalogen. For å få til det, må du gi app'en noen rettigheter. Se følgende referanse for hvordan du gjør det:

`http://developer.android.com/guide/webapps/webview.html`

## 6.3

- App'en i 6.2 og Notat02 går tilbake til forrige aktivitet når du trykker på tilbake-knappen. Skriv om en av disse programmet slik at tilbake-knappen fører deg tilbake til forrige webside i stedet.

- Du finner informasjonen du trenger ved å følge referansen under:

`http://developer.android.com/guide/webapps/webview.html#HandlingNavigation`

# Kapittel 7

# Oppsummering del 1

## 7.1 IP-adresse

### IPv4-adresse: 32 bit

- dot quad: fire 8-bits nummer
- eks.: 128.39.105.10

Listing 7.1: eksempler/01/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 aa fremskaffe ip-adressen til vert basert
10    * paa vertens domenenavn.
11    *
12    * Baade 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 aa 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 paa en annen maskin:"
```

```
33          );
34
35      String maskin = inngang.next();
36
37      /*
38       * For aa gjoere 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 gjoere et navneoppslag.
43       * (i DNS eller filen hosts).
44       *
45       *  Dersom 'vert' inneholder en tekststreng med ip-adresse,
46       *  gjoeres kun en sjekk paa om det er paa 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  }
```

## 7.2 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**

Listing 7.2: eksempler/01/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
23         utskriver.format(
24               "Interessant␣at␣du␣sier␣\"%s\".\n",
25               innleser.nextLine() );
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**

Listing 7.3: eksempler/01/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 paa port\t" +  datagramSocket.getLocalPort() );
15
16      datagramSocket.receive(datagram);
17
18      System.out.println(
19          "Datagrammet fra porten "    + datagram.getPort() +
20          " paa maskinen "              + datagram.getAddress() +
21          " er detektert.");
22
23      datagramSocket.close();
24    }
25  }
```

## 7.3 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

Listing 7.4: eksempler/03/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  }
```

Listing 7.5: eksempler/02/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
```

```
12    InterruptedException  {
13
14      opprettKonti();
15      opprettSpekulanter();
16
17      while (true){
18        skriv_saldo();
19        sleep(1000);
20      }
21    }
22
23    private static void opprettSpekulanter(){
24
25       int i;
26       for (i=0; i<S; i++)
27         new Balanse_3().start();
28    }
29
30    public void run()  {
31
32      while(true)
33        flyttPenger();
34    }
35
36    private static synchronized void flyttPenger() {
37
38      int til, fra, belop;
39
40      fra=Math.abs( slumptall.nextInt() ) %N;
41      til=Math.abs( slumptall.nextInt() ) %N;
42      belop=Math.abs( slumptall.nextInt() ) % ( (konto[fra]/10) );
43
44      konto[fra]-=belop;
45      konto[til]+=belop;
46    }
47
48    private static synchronized void opprettKonti() throws
49    InterruptedException{
50
51      int i;
52
53      for (i=0;i<N;i++)
54        konto[i]=T/N;
55
56      skriv_saldo();
57    }
58
59    private static synchronized void skriv_saldo() throws
60    InterruptedException{
61
62      int sum, i;
63
64      for (i=0, sum=0; i<N; i++){
65        System.out.printf("konto_%d:\t%d\n", i, konto[i]);
66        sum+=konto[i];
67      }
68
69      System.out.printf("----------\nTotalt:\t%d\n==========\n",sum);
```

```
70      System.out.flush();
71    }
72  }
```

## 7.4 Ikke-blokkerende-IO

Listing 7.6: eksempler/03/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 }
```

## 7.5 filer, URL-er kommandolinje, argumenter

Listing 7.7: losninger/03/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  }
```

## 7.6 RandomAccess-filtilgang, serialisering, vector

Listing 7.8: eksempler/04/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 }
```

Listing 7.9: eksempler/04/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  }
```

Listing 7.10: eksempler/04/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 }
```

## 7.7 HTML, CGi

### Miljøvariabler

Listing 7.11: eksempler/05/env.cgi

```
1  #!/bin/sh
2  java Env
```

Listing 7.12: eksempler/05/Env.java

```
1  public class Env {
```

```
2
3       public static void main(String[] args) {
4
5     System.out.printf("Content-type:text/plain\n\n");
6
7     for (java.util.Map.Entry<String, String> e : System.getenv().entrySet() )
8         System.out.println(e.toString());
9       }
10  }
```

## Html-skjema

Listing 7.13: eksempler/05/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>
```

Listing 7.14: eksempler/05/hallo_3.cgi

```
1  #!/bin/sh
2
3  java -Dfile.encoding=UTF-8 Hallo_3
```

Listing 7.15: eksempler/05/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 }
```

# 7.8 Android I

## Hva er Android?

- Komplett programvare-stack for mobile enheter.
- OS, mellomvare og nøkkelapplikasjoner

### Arkitektur

`http://developer.android.com/images/system-architecture.jpg`

## Applikasjons-komponenter

- Activity
- Services
- Broadcast receivers
- Content providers

## To måter å vis en webside i en android-applikasjon

### 1

Med en *Intent*

Listing 7.16: eksempler/notat01/src/ofa/notat01/Notat01.java

```
1  package ofa.notat01;
2
3  public class Notat01 extends android.app.Activity
4  {
5      @Override
6      public void onCreate(android.os.Bundle savedInstanceState)
7      {
8    super.onCreate(savedInstanceState);
9
10   startActivity(
11           new android.content.Intent(
12               android.content.Intent.ACTION_VIEW,
13               android.net.Uri.parse("http://oopva60.hive.no/04.html")
14               )
15           );
16     }
17 }
```

### 2

Med *WebView* satt i *ContentView*'et til *Activity*'en.

Listing 7.17: eksempler/notat02/src/ofa/notat02/Notat02.java

```
1  package ofa.notat02;
2
3  public class Notat02 extends android.app.Activity
4  {
5      /** Called when the activity is first created. */
6      @Override
7      public void onCreate(android.os.Bundle savedInstanceState)
8      {
9    super.onCreate(savedInstanceState);
10
11   android.webkit.WebView wv = new android.webkit.WebView(this);
12   setContentView(wv);
13
14   wv.getSettings().setJavaScriptEnabled(true);
15   wv.getSettings().setBuiltInZoomControls(true);
16   wv.loadUrl("file:///android_asset/04.html");
17      }
18 }
```

## 7.9 Øvelser

### 7.1

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.

### 7.2

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.

### 7.3

**Gi begrunnede eksempler på når det er naturlig å bruke**

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

### 7.4

En android-applikasjon kan laste en web-side ved bruk av *Intent* eller ved bruk av *WebView* i eget *ContentView*.

- Forklar hvordan disse metodene skiller seg fra hverandre og gi eksempler på anvendelser.

## 7.5

Resursser (html-dokumenten, bilder etc.) som brukes i en web-applikasjon for android-plattformen kan lagres *lokalt* på android-enheten eller *eksternt* på en web-server.

- Gi begrunnede eksempler på når det er naturlig å lagre henholdsvis internt og eksternt.

## 7.6

Hva menes med *serialisering*, og hvorfor gjøres det?

## 7.7

Hva må til for å kjøre et javaprogram med *Common Gateway Interface*?

## 7.8

Lag en android applikasjon som bruker et *WebView* til å vise frem navn og verdi på alle *miljøvariablene* til prosessen den kjører i.

**Tips:**

- Bygg opp hele siden som en tekststreng. Bruk denne tekststrengen som argument til loadData()-metoden til WebView'et.

- Se eksempler/05/Env.java for hvordan lese miljøvariablene.

# Del II

# Kapittel 8

# Servlets

## 8.1 Løsningsforslag på android-oppgaver

### 6.2

#### Oppgave

**a)** Lag en html-fil som inneholder noen hyperlenker til sider på weben.

**b)**

- Lag en android-applikasjon som bruker WebView til å vise frem html-fila fra a) og sidene den refererer til. Fila skal legges i assets-katalogen i prosjektkatalogen. For å få til det, må du gi app'en noen rettigheter. Se følgende referanse for hvordan du gjør det:

`http://developer.android.com/guide/webapps/webview.html`

#### Løsningsforslag

Listing 8.1: losninger/06/Losning_6_2/assets/6.2.a.html

```
1  <?xml version="1.0" encoding="UTF-8"?>
2
3  <!DOCTYPE
4      html PUBLIC "-//W3C//DTD XHTML Basic 1.1//EN"
5      "http://www.w3.org/TR/xhtml-basic/xhtml-basic11.dtd"
6      >
7
8  <html>
9
10   <head>
11
12     <title>Oppgave 6.2</title>
13     <meta
14         name='viewport'
15         content='width=device-width, user-scalable=yes, initial-scale=1.2'
16         />
17   </head>
18
19   <body>
```

```
20
21      <h1>Oppgave 6.2 </h1>
22
23      <ul>
24        <li>
25    <a href="http://www.hive.no">
26      hive
27    </a>
28        </li>
29
30        <li>
31    <a href="http://debbie.hive.no">
32      debbie
33    </a>
34        </li>
35
36        <li>
37    <a href="http://oopva60.hive.no">
38      oopva60
39    </a>
40        </li>
41
42      </ul>
43
44    </body>
45
46  </html>
```

Listing 8.2: losninger/06/Losning_6_2/AndroidManifest.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3        package="ofa.losning_6_2"
4        android:versionCode="1"
5        android:versionName="1.0">
6      <application android:icon="@drawable/icon" android:label="@string/
           app_name">
7          <activity android:name=".Losning_6_2"
8                    android:label="@string/app_name">
9              <intent-filter>
10                 <action android:name="android.intent.action.MAIN" />
11                 <category android:name="android.intent.category.LAUNCHER" /
                       >
12             </intent-filter>
13         </activity>
14
15     </application>
16     <uses-sdk android:minSdkVersion="8" />
17
18 <uses-permission android:name="android.permission.INTERNET"></uses-
       permission>
19 </manifest>
```

Listing 8.3: losninger/06/Losning_6_2/src/ofa/losning_6_2/Losning_6_2.java

```
1  package ofa.losning_6_2;
2
3  public class Losning_6_2 extends android.app.Activity
4  {
```

```
5       /** Called when the activity is first created. */
6       @Override
7       public void onCreate(android.os.Bundle savedInstanceState)
8       {
9           super.onCreate(savedInstanceState);
10
11          android.webkit.WebView wv = new android.webkit.WebView(this);
12          setContentView(wv);
13
14          wv.getSettings().setBuiltInZoomControls(true);
15
16          wv.setWebViewClient(new MinWebViewKlient());
17          wv.loadUrl("file:///android_asset/6.2.a.html");
18      }
19  }
```

Listing 8.4: losninger/06/Losning_6_2/src/ofa/losning_6_2/MinWebViewKlient.java

```
1  package ofa.losning_6_2;
2
3  import android.webkit.WebView;
4
5  public class MinWebViewKlient extends android.webkit.WebViewClient {
6
7
8    @Override
9    public boolean shouldOverrideUrlLoading(WebView view, String url) {
10
11     //return super.shouldOverrideUrlLoading(view, url);
12     return false;
13   }
14
15
16 }
```

## 6.3

### Oppgave

- App'en i 6.2 og Notat02 går tilbake til forrige aktivitet når du trykker på tilbake-knappen. Skriv om en av disse programmet slik at tilbake-knappen fører deg tilbake til forrige webside i stedet.

- Du finner informasjonen du trenger ved å følge referansen under:

`http://developer.android.com/guide/webapps/webview.html#HandlingNavigation`

### Løsningsforslag

Listing 8.5: losninger/06/Losning_6_3/src/ofa/losning_6_3/Losning_6_3.java

```
1  package ofa.losning_6_3;
2
3  public class Losning_6_3 extends android.app.Activity
4  {
5
6    android.webkit.WebView wv;
7
```

```
8       @Override
9       public void onCreate(android.os.Bundle savedInstanceState)
10      {
11          super.onCreate(savedInstanceState);
12
13          wv = new android.webkit.WebView(this);
14          setContentView(wv);
15
16          wv.getSettings().setBuiltInZoomControls(true);
17
18          wv.setWebViewClient(new MinWebViewKlient());
19          wv.loadUrl("file:///android_asset/6.3.html");
20      }
21
22      @Override
23      public boolean onKeyDown(int keyCode, android.view.KeyEvent event) {
24
25        if (
26            keyCode == android.view.KeyEvent.KEYCODE_BACK &&
27            wv.canGoBack() ){
28
29          wv.goBack();
30          return true;
31        }
32        return super.onKeyDown(keyCode, event);
33      }
34  }
```

## 8.2 Om servlets

Servlets er javaklasser som følger *Java Servlet API*'et. I likhet med CGI-programmer er *servlets* er programmer som (normalt) kjører på en web-tjener. Dette står i motsetning til en *applet* som kjøres på klienten.

- Kjøringen trigges av http-forespørsel fra web-klient.
- Produserer dokument (f.eks. et html-dokument) som returneres til klienten.

En *servletcontainer* laster og start servlets, håndterer hele dens livssyklus og kommunikasjon med webserveren. Ofte er web-server og serlvetcontainer i samme programpakke.

- Servlets kjøres i separate tråder, men i samme prosess som servlet-containeren. Dette gir noen vesentlige fordeler framfor CGI-programmer.

### En servlets livs-syklus:

init()

- kjøres når servlet-containeren starter en servlet.

service()

- Hele levetiden til servlet'en venter den på http-forespørsel fra klienten.
- Kaller doGet(), doPost(), doPut() eller doDelete() avhengig av forespørsels-typen.

**destroy()**

Kjøres når servleten stoppes. Eksempler på servlet-containere:
Apache Tomcat
GlassFish
Jetty

## 8.3 Jetty

`http://wiki.eclipse.org/Jetty/Starting`

- Skriptet under viser hvordan jetty kan installeres og startes.

Listing 8.6: eksempler/08/install-jetty.sh

```
#!/bin/sh

# Finn versjons nummer paa nettsiden
JETTY_VERSION=7.3.0.v20110203

# Laster ned
wget http://download.eclipse.org/jetty/$JETTY_VERSION/\
dist/jetty-distribution-$JETTY_VERSION.tar.gz

# Pakker opp
tar -xzf jetty-distribution-$JETTY_VERSION.tar.gz

echo
echo For aa starte jetty:
echo ----------------------
echo $ cd jetty-distribution-$JETTY_VERSION
echo $ java -jar start.jar
echo ------------------------
```

## 8.4 Eksempler

### ServletHallo

Listing 8.7: eksempler/08/ServletHallo.java

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

  // gjoer en http-get
    public void doGet(
        javax.servlet.http.HttpServletRequest  request,
        javax.servlet.http.HttpServletResponse respons )

        throws java.io.IOException,
               javax.servlet.ServletException {

      // http-hodet
      respons.setContentType("text/Plain");

      java.io.PrintWriter ut = respons.getWriter();
```

```
15
16          // http-kroppen
17          ut.println("Hallo");
18            ut.flush();
19        }
20  }
```

```
javac -cp .:/usr/share/java/servlet-api.jar ServletHallo.java

servlethallo/WEB-INF/classes/ServletHallo.class
servlethallo/WEB-INF/web.xml
```

Listing 8.8: eksempler/08/servlethallo/WEB-INF/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>
```

**war-filer**

- web-archive

```
jar -cf ServletHallo.war .

cp ServletHallo.war $JETTY_HOME/webapp/
```

# $JETTY_HOME er den katalogen som jetty ble pakket ut i (jetty-distrib...)

- 'exploding' og utrulling (deployment) utføres automatisk av Jetty.

## Input fra html-skjema

Listing 8.9: eksempler/08/skjemaservlet/skjemaservlet.html

```
1  <html>
2  <form action="SkjemaServlet">
3  Brukernavn: <input type=text name=Brukernavn> <p>
4  <input type=submit>
5  </form>
6  </html>
```

Listing 8.10: eksempler/08/SkjemaServlet.java

```
1  public class SkjemaServlet  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        java.io.PrintWriter ut = respons.getWriter();
10
11         String brukernavn = request.getParameter("Brukernavn");
12
13   respons.setContentType("text/Plain");
14         ut.println("Hallo "+brukernavn+"!");
15
16         ut.flush();
17     }
18 }
```

## Informasjonskapsler

Listing 8.11: eksempler/08/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 || kake.length == 0) {
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.printf("Takk for sist! Du har %d kake(r)<p>", kake.length );
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  }
```

## Sesjoner

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

Listing 8.12: eksempler/08/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 }
```

## 8.5 Øvelser

### 8.1

- Lag en servlet som ved første besøk fra en bruker gjør følgende: Henter brukernavn, passord, fornavn, etternavn fra et html-skjema. Ved senere besøk, hentes informasjonen frem og vises frem for brukeren.

#### a)

Løs dette ved å lagre dataene som informasjonskapsler (cookies) i klienten.

#### b)

Løs dette ved å lagre dataene sesjonsobjektet på i tjeneren.

- For å teste servlet'ene på debbie, kan dere køre jetty.

Pass på at default portnummer (8080) kan være opptatt. Portnummeret kan endres i *etc/jetty.xml*. Prøv f.eks. å bruk din brukerid (*echo $UID*) som portnummer.

### 8.3

- i-jetty er en variant av servlet-containeren for android-plattformen. Denne oppgaven går ut på installer i-jetty på en android-enhet (fysisik eller virtuell) og få løsningen i 8.1 til å kjøre på den.

**Web-siden til i-jetty:**

`http://code.google.com/p/i-jetty/`

**Instruksjoner for å lage war-filer for i-jetty:**

`http://code.google.com/p/i-jetty/wiki/DownloadableWebapps`

# Kapittel 9

# JavaServer Pages

## 9.1 Løsningsforslag

### 8.1

**Oppgave**

- Lag en servlet som ved første besøk fra en bruker gjør følgende: Henter brukernavn, passord, fornavn, etternavn fra et html-skjema. Ved senere besøk, hentes informasjonen frem og vises frem for brukeren.

**a)** Løs dette ved å lagre dataene som informasjonskapsler (cookies) i klienten.

**b)** Løs dette ved å lagre dataene sesjonsobjektet på i tjeneren.

- For å teste servlet'ene på debbie, kan dere køre jetty.

Pass på at default portnummer (8080) kan være opptatt. Portnummeret kan endres i *etc/jetty.xml*. Prøv f.eks. å bruk din brukerid (*echo $UID*) som portnummer.

**Løsning**

Listing 9.1: losninger/08/Losning8_1a.java

```
1  public class Losning8_1a extends javax.servlet.http.HttpServlet {
2
3    private static final long serialVersionUID = 1L;
4
5    public void doGet(
6        javax.servlet.http.HttpServletRequest  request,
7        javax.servlet.http.HttpServletResponse respons )
8
9    throws
10   java.io.IOException,
11   javax.servlet.ServletException {
12
13     javax.servlet.http.HttpSession session = request.getSession();
14     java.io.PrintWriter ut = respons.getWriter();
15
16     respons.setContentType("text/html");
17
```

```
18      ut.println("<html><body>");
19
20      String[] felt = {"brukernavn", "passord", "fornavn", "etternavn"};
21
22      if (session.isNew()){
23
24        ut.println("<form>");
25
26        for (String f:felt)
27          ut.printf("%s: <input type=text name='%s'><br>",f,f);
28
29        ut.println("<input type=submit name='skjema' value='sendt'></form>");
30
31      }
32      else if (request.getParameter("skjema")!=null){
33
34        for (String f:felt)
35          respons.addCookie(
36              new javax.servlet.http.Cookie( f, request.getParameter(f))
37          );
38
39        ut.println("Takk :) <a href='Losning8_1a'>Fortsett</a>");
40      }
41
42      else {
43        javax.servlet.http.Cookie[] kake = request.getCookies();
44
45        for (int i=0; i<kake.length;i++)
46          ut.printf(
47              "%s: %s<br>",
48              kake[i].getName(),
49              kake[i].getValue()
50          );
51
52      }
53
54      ut.println("</body></html>");
55      ut.flush();
56    }
57  }
```

Listing 9.2: losninger/08/Losning8_1b.java

```
1   public class Losning8_1b extends javax.servlet.http.HttpServlet {
2
3     private static final long serialVersionUID = 1L;
4
5     public void doGet(
6         javax.servlet.http.HttpServletRequest  request,
7         javax.servlet.http.HttpServletResponse respons )
8
9     throws
10    java.io.IOException,
11    javax.servlet.ServletException {
12
13      javax.servlet.http.HttpSession session = request.getSession();
14      java.io.PrintWriter ut = respons.getWriter();
```

```
15
16      respons.setContentType("text/html");
17
18      ut.println("<html><body>");
19
20      String[] felt = {"brukernavn", "passord", "fornavn", "etternavn"};
21
22      if (session.isNew()){
23
24        ut.println("<form>");
25
26        for (String f:felt)
27          ut.printf("%s: <input type=text name='%s'><br>",f,f);
28
29        ut.println("<input type=submit name='skjema' value='sendt'></form>");
30
31      }
32      else if (request.getParameter("skjema")!=null){
33
34        for (String f:felt)
35          session.setAttribute(f, request.getParameter(f));
36
37        ut.println("Takk :) <a href='Losning8_1b'>Fortsett</a>");
38      }
39
40      else
41        for (String f:felt)
42          ut.printf("%s: %s<br>",f,session.getAttribute(f));
43
44          ut.println("</body></html>");
45      ut.flush();
46    }
47  }
```

## 9.2 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:

Listing 9.3: eksempler/09/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>
```

## 9.3 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)

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 innheholder tagger for iterasjoner, betingelser, XML, internasjonalisering, DB-access (m/SQL) og et get språk tilpasset web-designere/utviklere:*Expression Language* (EL).

## 9.4 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

### 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.
- En vanlig "arbeidsfordeling" mellom applikasjonskomponenter:
- 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

## 9.5 Kompilering og kjøring

- JSP-filene ligger på samme katalog som html-filene.
- En web-tjener trenger en servlet-container for å håndtere servlets, og en JSP-container å håndtere JSP'er. (Vi bruker Jetty som er både web-tjener, servlet-container og JSP-container).
- JSP blir kompilert til servlets ved første klientforespørsel.
- Den kompilerte filen beholdes inntil tjeneren terminerer eller JSP-kildekoden endres.
- JSP-containeren avbryter http-forespørsel om en JSP og videresender forespørselen til den servlet'en som er knyttet til JSP'en.
- Kompileringen fører til lang respostid ved første referanse. Prekompilering kan gjøres enten ved å gjøre en vanlig referanse, eller ved bruk av det boolske parameteret 'jsp_precompile'. Dette parameteret har defaultverdien 'true'. (Eks.: http://oopva60.hive.no:8080/

### Siden JSP kompileres til servlet, har de mange av de samme fordelene som 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"

## 9.6 Ulike elementer

### To typer elementer

**1: template text**

- Blir alltid sendt rett gjennom til nettleseren

**Statisk innhold**

- HTML
- plain-text
- XML
- etc.

**2: JSP-elementer**

noen få JSP-elementer for dynamisk innhold

- Ved forespørsel blir template-tekst og JSP-elementer slått sammen

## 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:

Listing 9.4: eksempler/09/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>
```

## 9.7 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 objektertilgjengelig for mer enn en brukerf.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"

## 9.8 Tilgang til DB via JDBC - tre måter

### 1. plassere nødvendig kode i JSP'en

**Vi ser på et eksempel:**

Listing 9.5: eksempler/09/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://debbie.hive.no/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>
```

`http://docs.codehaus.org/display/JETTY/Classloading`

### 2. definere spesialtilpassede "tags"

(ikke pensum)

### 3. bruke JavaBeans

vi ser eksempel på dette i neste forelesning

## 9.9 To hovedmetoder for samarbeid mellom servlets og JSP

### 1. Applikasjonens miljø

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

### 2. Brukernes sesjon

- tilgjengelig i servlet som 'Session'-objekt
- tilgjengelig i JSP via det implisitte objektet 'session'
- Vi ser på et eksempel.

### Eksempel på samhandling mellom servlet og JSP

- Under ser vi en servlet og en JSP. Begge viser en verdi på attributtet 'teller' og sesjonsidentifikatoren. Servlet'en øker 'teller's verdi med en.

Listing 9.6: eksempler/09/SesjonsTest2.java

```
1  public class SesjonsTest2  extends javax.servlet.http.HttpServlet {
2
3    private static final long serialVersionUID = 1L;
4
5    public void doGet(
6        javax.servlet.http.HttpServletRequest  request,
7          javax.servlet.http.HttpServletResponse respons )
8
9          throws java.io.IOException,
10         javax.servlet.ServletException {
11
12       java.io.PrintWriter ut;
13       Integer teller;
14
15       javax.servlet.http.HttpSession session = request.getSession();
16       teller = (Integer) session.getAttribute("teller");
17        if (teller==null)
18            teller = new Integer(1);
19        else
20            teller = new Integer(teller.intValue() + 1);
21        session.setAttribute("teller", teller);
22
23       ut = respons.getWriter();
24       respons.setContentType("text/html");
25
```

```
26          ut.printf("Sesjonens identifikator (cookie med navnet JSESSIONID):<BR
                >%s<P>", session.getId() );
27          ut.printf("Sesjonens Integer med navn 'teller':<BR>%d<P>", teller);
28          ut.println("<A HREF=Sesjonsdata.jsp> Hyperlenke til JSP </A>");
29          ut.flush();
30      }
31  }
```

Listing 9.7: eksempler/09/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>
```

## 9.10 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:

Listing 9.8: eksempler/09/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>
```

Listing 9.9: eksempler/09/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://debbie.hive.no/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>
```

## 9.11 Øvelser

### 9.1

Løs oppgave 8.1 med JSP'er i stedet for som servlets.

## 9.2

Forandre JSP'en Sesjonsdata.jsp (som er vist over) slik at det også øker verdien 'teller' hver gang den blir forespurt.

## 9.3

### a)

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

Listing 9.10: eksempler/09/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).

# Kapittel 10

# JavaBeans

## 10.1 Løsning på tidligere oppgave

### 9.1

- Løs oppgave 8.1 med JSP'er i stedet for som servlets.

Listing 10.1: losninger/09/Losning9_1a.jsp

```
1  <html>
2  <head>
3    <title>CookieJSP</title>
4  </head>
5  <body>
6
7    <%! String[] felt = {"brukernavn", "passord", "fornavn", "etternavn"}; %>
8
9
10   <% if (session.isNew()){ %>
11
12   <form>
13
14   <%  for (String f:felt)
15         out.println(f+": <input type=text name='"+f+"'><br>"); %>
16
17   <input type=submit name='skjema' value='sendt'>
18   </form>
19
20   <%  }
21     else if (request.getParameter("skjema")!=null){
22
23       for (String f:felt)
24         response.addCookie(
25             new javax.servlet.http.Cookie( f, request.getParameter(f))
26         ); %>
27
28   Takk :) <a href='Losning9_1a.jsp'>Fortsett</a>
29
30
31   <%  }
32
33     else {
34
```

```
35        javax.servlet.http.Cookie[] kake = request.getCookies();
36
37        for (int i=0; i<kake.length;i++)
38          out.println(
39               kake[i].getName()+": "+kake[i].getValue()+"<br>"
40          );
41
42      }
43  %>
44
45  </body>
46  </html>
```

Listing 10.2: losninger/09/Losning9_1b.jsp

```
1   <html>
2   <head>
3     <title>SesjonsJSP</title>
4   </head>
5   <body>
6
7     <%! String[] felt = {"brukernavn", "passord", "fornavn", "etternavn"}; %>
8
9
10    <% if (session.isNew()){ %>
11
12    <form>
13
14    <%  for (String f:felt)
15          out.println(f+": <input type=text name='"+f+"'><br>"); %>
16
17    <input type=submit name='skjema' value='sendt'>
18    </form>
19
20    <%  } else if (request.getParameter("skjema")!=null) {
21
22          for (String f:felt)
23            session.setAttribute(f, request.getParameter(f));
24            %>
25
26    Takk :) <a href='Losning9_1b.jsp'>Fortsett</a>
27
28
29    <%  } else
30          for (String f:felt)
31            out.println(f+": "+session.getAttribute(f)+"<br>");
32            %>
33
34  </body>
35  </html>
```

## 9.3

### a)

Gjør om SqlKlient2 fra et cgi-program til JSP.

### b)

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

Listing 10.3: losninger/09/sqlklient.jsp

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

<HTML>
<HEAD>
  <TITLE>DB-connect</TITLE>
</HEAD>
<BODY>

<%
Class.forName("com.mysql.jdbc.Driver");
Connection lnk = DriverManager.getConnection(
  "jdbc:mysql://localhost/bokbase",
  "db",
  "ada"
);
%>

<form>
 <input
  type=text
  size=100
  name=s value='<%= request.getParameter("s")%>'
 />
</form>
<table border=1>
<tr>
<%
ResultSetMetaData met;
ResultSet res;
int kol, i;

res = lnk.createStatement().executeQuery(request.getParameter("s"));
met = res.getMetaData();
kol = met.getColumnCount();
out.print(kol);

 for (i=1; i <= kol; i++)
   out.print(met.getColumnName(i));
%>
</tr>

<% while(res.next()){ %>

 <tr>

<% for ( i = 1; i <= kol; i++ ) {%>

  <td> <%= res.getString(i) %> </td>

<%   }%>
</tr>
<%}%>

```

```
54 </table>
55 </BODY>
56 </HTML>
```

## 10.2 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
```

## 10.3 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'

#### Attributtet 'scope' angir tilgjengelighet

- page (default)
- request
- session
- application (globalt for alle brukere)

**Eksempel:**

<jsp:useBean id="bonne" class="bonner.HalloBonne" scope="request"/>

## 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="*"

## Expression language

- tilgjengelig fra JSP 2.0
- gir enklere syntax.
- (ikke på pensum)

`http://download.oracle.com/javaee/1.4/tutorial/doc/JSPIntro7.html`

## 10.4 Eksempler

### Hallo

Listing 10.4: eksempler/10/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 }
```

Listing 10.5: eksempler/10/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>
```

### Person 1

Listing 10.6: eksempler/10/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  }
```

Listing 10.7: eksempler/10/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>
```

## Person 2

Listing 10.8: eksempler/10/person2.jsp

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

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

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

<jsp:setProperty name="pB" property="*" />

<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>
```

## JDBC

Listing 10.9: eksempler/10/bonner/SqlBonne.java

```
package bonner;

public class SqlBonne implements java.io.Serializable {

  private static java.sql.ResultSetMetaData met;
  private static java.sql.Statement          stm;

  private String sql = "show tables";

  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://oopva60.hive.no/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 }
```

Listing 10.10: eksempler/10/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>
```

## 10.5 Øvelser

### 10.1

#### a)

Lag en bønne som inneholder heltalls-variabelen *teller*. Bønna skal ha *setter* og *getter* for teller. I tillegg skal den ha en metode for å *øke* teller.

#### b)

Lag en JSP som bruker bønna i a) til å holde telling med hvor mange http-forspørsler som er gjort i en pågående sesjon.

### 10.2

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 11

# 11 Android og JavaScript

## 11.1 Løsningsforslag fra forrige gang

### 10.1 a)

#### Oppgavetekst

Lag en bønne som inneholder heltalls-variabelen *teller*. Bønna skal ha *setter* og *getter* for teller. I tillegg skal den ha en metode for å *øke* teller.

#### Løsningsforslag

Listing 11.1: losninger/10/WEB-INF/classes/tellerbonne/TellerBonne.java

```
1  package tellerbonne;
2
3  public class TellerBonne {
4
5    private int teller = 0;
6
7    public int getTeller() {
8      return teller;
9    }
10
11   public void setTeller(int teller) {
12     this.teller = teller;
13   }
14
15   public void increaseTeller() {
16     this.teller ++;
17   }
18 }
```

### 10.1 b)

#### Oppgavetekst

Lag en JSP som bruker bønna i a) til å holde telling med hvor mange http-forspørsler som er gjort i en pågående sesjon.

#### Løsningsforslag

Listing 11.2: losninger/10/bonnesesjonsteller.jsp

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<%@ page errorPage="Feilmelding.jsp" %>

<html>
<head>
  <title> Teller transaksjoner pr. sesjon v.h.a. bønne</title>
</head>
<body>


<jsp:useBean id="tb" class="tellerbonne.TellerBonne" scope="session" />

<% tb.increaseTeller(); %>
Antall gjennomførte transaksjoner i denne sesjonen:
<jsp:getProperty name="tb" property="teller" />

</body>
</html>
```

## 11.2 Rotasjon

- Ved rotasjon av en android-applikasjon kjøres metoden onCreate() hos den aktive Aktiviteten.
- I de fleste tilfeller ønsker vi å beholde applikasjonens tilstand ved rotasjon

### to teknikker

- bundle og lagre tilstand
- manifest - ignorere

### notat4-eksempel

Listing 11.3: eksempler/11/notat04/src/ofa/notat04/Notat04.java

```
package ofa.notat04;

/*
 *  To teknikker for aa beholde tilstand ved rotasjon av webview
 *  Den ene finnes i manifestet,
 *  den andre bruker web-viewets saveState() og restoreState()
*/

import android.os.Bundle;

public class Notat04 extends android.app.Activity {

  android.webkit.WebView wv;

    @Override
    public void onCreate(android.os.Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);

```

```
19        wv = new android.webkit.WebView(this);
20        setContentView(wv);
21
22        if (savedInstanceState==null){
23          wv.getSettings().setJavaScriptEnabled(true);
24          wv.getSettings().setBuiltInZoomControls(true);
25          wv.loadUrl("file:///android_asset/04.html");
26        }
27        else
28          wv.restoreState(savedInstanceState);
29
30      }
31
32      @Override
33      public boolean onKeyDown(int keyCode, android.view.KeyEvent event) {
34
35        if (
36        keyCode == android.view.KeyEvent.KEYCODE_BACK &&
37        wv.canGoBack() ){
38
39        wv.goBack();
40        return true;
41        }
42
43    return super.onKeyDown(keyCode, event);
44      }
45
46    @Override
47    protected void onSaveInstanceState(Bundle outState) {
48      //super.onSaveInstanceState(outState);
49      wv.saveState(outState);
50    }
51
52  }
```

Listing 11.4: eksempler/11/notat04/AndroidManifest.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3        package="ofa.notat04"
4        android:versionCode="1"
5        android:versionName="1.0">
6      <application android:icon="@drawable/icon" android:label="@string/
           app_name">
7          <activity android:name=".Notat04"
8                    android:label="@string/app_name"
9                    android:configChanges="keyboardHidden|orientation">
10             <intent-filter>
11                 <action android:name="android.intent.action.MAIN" />
12                 <category android:name="android.intent.category.LAUNCHER" /
                       >
13             </intent-filter>
14         </activity>
15     </application>
16     <uses-sdk android:minSdkVersion="8" />
17     <uses-permission android:name="android.permission.INTERNET" />
18 </manifest>
```

## 11.3 javascript koblet med java objekter

### fra java-objekt til html-side

Listing 11.5: eksempler/11/JavaScriptTest/src/ofa/javascripttest/JavaScriptGrensesnitt.java

```
1  package ofa.javascripttest;
2
3  import android.widget.Toast;
4
5  public class JavaScriptGrensesnitt {
6
7    android.content.Context kontekst;
8
9    public JavaScriptGrensesnitt(android.content.Context kontekst) {
10     super();
11     this.kontekst = kontekst;
12   }
13
14   public void visToast(String tekst) {
15     Toast.makeText(kontekst, tekst, android.widget.Toast.LENGTH_SHORT).show
           ();
16   }
17 }
```

Listing 11.6: eksempler/11/JavaScriptTest/src/ofa/javascripttest/JavaScriptTest.java

```
1  package ofa.javascripttest;
2
3  public class JavaScriptTest extends android.app.Activity {
4
5    android.webkit.WebView wv;
6
7      @Override
8      public void onCreate(android.os.Bundle savedInstanceState)
9      {
10         super.onCreate(savedInstanceState);
11
12         wv = new android.webkit.WebView(this);
13
14         wv.loadUrl("file:///android_asset/index.html");
15         wv.getSettings().setJavaScriptEnabled(true);
16         wv.addJavascriptInterface(
17             new JavaScriptGrensesnitt(this),
18             "javaobjekt"
19         );
20         setContentView(wv);
21     }
22  }
```

Listing 11.7: eksempler/11/JavaScriptTest/assets/index.html

```
1  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
2  <html>
3    <head>
4      <meta name='viewport'
5      content='width=device-width, user-scalable=yes, initial-scale=1.2'
6      />
7      <meta http-equiv="Content-type" content="text/html;charset=UTF-8">
```

```
8       <title>webapptest</title>
9     </head>
10
11    <body>
12      <button onclick="javaobjekt.visToast('BÃ¸')">
13        Klikk meg!
14      </button>
15    </body>
16
17  </html>
```

## data fra html-skjema til java-objekter (og tilbake)

Listing 11.8: eksempler/11/JavaScriptHtmlFormInput/assets/index.html

```
1  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
2  <html>
3    <head>
4      <meta name='viewport'
5      content='width=device-width, user-scalable=yes, initial-scale=1.2'
6      />
7      <meta http-equiv="Content-type" content="text/html;charset=UTF-8">
8      <title>Form input</title>
9    </head>
10
11   <body>
12     <form name='skjema'>
13       <input type=text name='tekstfelt'>
14     </form>
15
16   <button onclick="alert(skjema.tekstfelt.value)">
17       javascript-alert
18     </button>
19
20     <button onclick="javaobjekt.visToast(skjema.tekstfelt.value)">
21       javaobjekt
22     </button>
23
24   <button onclick="ut.innerHTML=skjema.tekstfelt.value">
25       html-element
26     </button>
27
28   <button onclick="javaobjekt.skrivUt(skjema.tekstfelt.value)">
29       javaobjekt til html-element
30     </button>
31
32   <pre id='ut'>
33   test
34   </pre>
35   </body>
36
37 </html>
```

Listing 11.9: eksempler/11/JavaScriptHtmlFormInput/src/ofa/forminput/FormInput.java

```
1  package ofa.forminput;
2
3  public class FormInput extends android.app.Activity {
4
```

```
5    android.os.Handler mHandler = new android.os.Handler();
6    android.webkit.WebView wv;
7    android.content.Context kontekst = this;
8    MinWebChromeKlient wcc = new MinWebChromeKlient();
9
10   @Override
11   public void onCreate(android.os.Bundle savedInstanceState)
12   {
13     super.onCreate(savedInstanceState);
14
15     wv = new android.webkit.WebView(this);
16
17     wv.getSettings().setJavaScriptEnabled(true);
18     wv.addJavascriptInterface(wcc, "javaobjekt");
19     wv.setWebChromeClient(wcc);
20     wv.loadUrl("file:///android_asset/index.html");
21
22     setContentView(wv);
23   }
24
25   class MinWebChromeKlient extends android.webkit.WebChromeClient {
26
27     @Override
28     public boolean onJsAlert(
29         android.webkit.WebView view,
30         String url,
31         String message,
32         android.webkit.JsResult result) {
33
34       this.visToast(message);
35       result.confirm();
36       return true;
37     }
38
39     public void visToast(String tekst) {
40
41       android.widget.Toast.makeText(
42           kontekst,
43           tekst,
44           android.widget.Toast.LENGTH_SHORT
45       ).show();
46     }
47
48     public void skrivUt(final String tekst) {
49
50       mHandler.post(
51           new Runnable() {
52             public void run() {
53
54               wv.loadUrl("javascript:ut.innerHTML='"+tekst+"'");
55             }
56           });
57     }
58   }
59 }
```

## 11.4 Krasj-kurs i Javascript

Listing 11.10: eksempler/11/javascripthallo.html

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

  <head>
    <meta name='viewport'
    content='width=device-width, user-scalable=yes, initial-scale=1.2'
    />
    <meta http-equiv="Content-type" content="text/html;charset=UTF-8">
    <title>JavaScript-hallo</title>
  </head>

  <body>

    <script type="text/javascript">
      document.write("<h1 id='hilsen'>Hallo</h1>");
      overskrift = document.getElementById('hilsen');
    </script>

    <button onclick="overskrift.innerHTML='Borte'">
      Klikk her
    </button>

  </body>

</html>
```

## Leksikal struktur

- Javascript skiller mellom store og små bokstaver.
- Hendelseshåndterere som er definert som et html-attributt er skiller ikke på store og små bokstaver.
- Det er vanlig å skrive slike med en blanding av små og store bokstaver. F.eks. slik:
- onClick
- eller slik:
- OnClick
- For å referere til hendelsen i javascript-kode må den navngies med små bokstaver, slik:
- onclick
- Kommandosetninger (en: Statements) avsluttes med semikolon eller linjeskift.
- Kommentarer skrives i koden som i C, C++ og Java.
- <!– tolkes som //
- –> blir ikke gjenkjent.

## Tall

- Alle tall er representert som flyttall.

**Oktale tall angis med en null foran. F.eks. slik:**

034

**Hexadesimale tall angis med 0x eller 0X foran. F.eks. slik:**

- 0x4f
- 0X3E
- 0xA1
- 4.89
- 1.432-e5
- 8.9E+13

**I tillegg til vanlige aritmetiske operatorer (+,-,/,*), ligger mange matematiske funksjoner i objektet 'Math'. Eksempler:**

- Math.sin(13)
- Math.sqrt(4)

**Metoden 'toString' kan brukes for å gjøre om tall til tekst: (123).toString**

- x.toString;
- x.toString(16); //hexadesimal

**Spesielle tall**

- Number.MAX_VALUE
- Number.MIN_VALUE
- Number.POSITIVE_INFINITY // > Number.MAX_VALUE
- Number.NEGATIVE_INFINITY // < Number.MIN_VALUE

**Number.NaN - Not a Number**

- returneres dersom en regneoperasjon feiler. (F.eks. hvis man deler på null.)
- beregnes ikke som lik noen annen verdi.
- bruk funksjonen 'isNaN()' for å teste på dette tallet.

## Tekststrenger

- ”Bakoverskråstrek-tegn” - som i C,C++ og java. (nesten hvertfall)
- Indeksering starter på null som i C,C++ og java.
- Eksempler på tekstmanupulerings-metoder:
- var tekst=”bla bla”
- var tekstlengde=tekst.lenght
- tekst.substring(1,4)
- tekst.indexOf('l')

## Boolske verdier

- 1 == 1 // returnerer true
- 2 == 1 //returnerer false

## Funksjoner

function testfunk(x) { return ”Funksjonen fikk dette argumentet: ”+x }

## Rekker

var rekke1=[1,,,,9]

## 'null'

- er ikke det samme som tallet 0.
- betyr 'uten verdi'

## 'Undefined'

**'Undefined' er verdien til:**

- variabel som ikke eksisterer
- variabel som ikke er initialisert

**Følgende er sant:**

undefined==null

## 1.7 JavaScript i Nettlesere

### 1.7.1 Objekthierariket

- I klientside-programmer kalles det globale objektet Window. Dette representerer et vindu eller en ramme
- (en: frame). Globale variabler er derfor synonymt med egenskaper til 'Window-objektet'.

#### 1.7.1.1 window.

- self,window,parent,top vindus objekter
- navigator nettleser objekt (se egen liste)
- frames[]|] rekke av vindusobjekter
- location URL-en til dokumentet som vises
- history 'surfe-historien' (forrige side etc.)
- document html-dokumentet (se egen liste)

#### 1.7.1.2 window.navigator.

- plugins[]|] rekke av plugin-objekter (f.o.m Javascript 1.1)
- mimeTypes[]|] rekke av mimetype-objekter (f.o.m Javascript 1.1)

#### 1.7.1.3 window.document.

- forms[]|] rekke av skjemaobjekter (se egen liste)
- anchors[]|] rekke av anker-objekter
- links[]|] rekke av hyperlenke-objekter
- images[]|] rekke av bilde-objekter (f.o.m Javascript 1.1)
- applets[]|] rekke av applet-objekter (f.o.m Javascript 1.1)
- embeds[]|] rekke av innbakte objekter (f.o.m Javascript 1.1)

#### 1.7.1.4 window.document.forms[][].

- elements[]|] rekke av skjemaelement-objekter (knapper, tekstfelt, etc.)
- elements[]|].options[]|] rekke av egenskaper hos skjema-elementene

### 1.7.2 JavasScript i HTML

- JavaScript-kode kan settes inn i et html-dokument, ved å omslutte den med taggen <SCRIPT></SCRIPT>.
- Denne taggen kan stå både html-dokumentets, hode og kropp.
- Selv om det kan være mange slike tagger i et dokument, anses de tilsammen å utgjøre ett JavaScriptprogram.
- Dersom javascript koden skal skrive '</SCRIPT' til et annet vindu eller ramme, må
- det gjøres litt spesielt slik at ikke scriptet som kjører, avsluttes. Det kan løses ved at tekststrengen skrives
- ut i to deler. F.eks. slik:
- <SCRIPT> ... vindu1.document.write("<SCRIPT>document.write('hei')</" +"SCRIPT>") ... <SCRIPT>

#### 1.7.2.1 LANGUAGE

- Ved å bruke attributtet LANGUAGE til taggen SCRIPT, kan man spesifisere hvilket språk og hvilken
- versjon av språket som skriptet er skrevet i. Dersom man utelater dette attributtet, vil både Navigator
- og Explorer, regne med at det er JavaScript som brukes.
- Eksempler:
- <SCRIPT LANGUAGE=JavaScript> ... kode kommer her ... <SCRIPT>
- <SCRIPT LANGUAGE=JavaScript1.1> ... kode kommer her ... <SCRIPT>
- <SCRIPT LANGUAGE=JavaScript1.2> ... kode kommer her ... <SCRIPT>

#### 1.7.2.2 SRC

- F.o.m. JavaScript 1.1 kan attributtet SRC brukes for å inkludere kode som ligger i en annen fil. Filen
- angis som en URL. Slike filer innholder ren kode, uten html-koder.
- De har vanligvis filendelsen .js.
- Vevtjeneren skal være satt opp slik at filendelsen er assosiert med MIME-typen application/x-javascript.

- Nettlesere som kjenner attributtet, ignorerer eventuell kode som står mellom <SCRIPT SRC=...> og
- </SCRIPT>.
- Nettlesere som ikke kjenner attributtet, utfører eventuell kode som står mellom <SCRIPT SRC=...>
- og </SCRIPT>.
- Eksempler:
- <SCRIPT SRC="../../mineskript/diverse.js"><SCRIPT>

`http://tulla.hivert.no/mineskript/diverse.js`

#### 1.7.2.3 ARCHIVE

- Man kan pakke sammen flere JavaScript filer (og andre filer; f.eks. signaturer) i JAR(Java-arkiv)-filer.
- Netscape tilbyr et gratis program for lage slike filer.
- Eksempel:
- <SCRIPT
- ARCHIVE="diverse.jar"
- SRC="kalkulasjoner.js">
- <SCRIPT>

#### 1.7.2.4 Hendelseshåndterere

- Hendelseshåndterere er definert som attributter til html-tagger.
- Verdien til attributtet er koden som skal utføres.
- Det er vanlig at koden kun består av et funksjonskall, slik at det blir lettere å lese.
- Eksempel:
- <INPUT TYPE = "submit" name = "send" value = "Send" onClick = "skjemaKontroll();" >

#### 1.7.2.5 Pseudo-protokollen javascript:

- Ved å bruke pseudo-protokollen javascript: vil en URL'en bli tolket som JavaScript-kode.
- Når nettleseren laster en slik URL, vil koden bli kjørt.
- Tekststreng-verdien av den siste kommando-setningen vil utgjøre dokumentet som vises.
- Dersom den siste kommando-setningen ikke har noen tekststreng-verdi, vil ikke dokumentet som allerede er lastet i nettleseren endres. F.o.m JavaScript 1.1, kan man fremtvinge en slik situasjon v.h.a. operatoren void.
- Ved å bruke slike URL'er i hypertekst-lenker eller som verdi på ACTION-attributtet til htmlskjema, blir resultatet en slags hendelseshåndterer.
- I Navigator kan man skrive 'javascript:-URL'er' rett inn i 'Location-feltet'.
- Dersom man i Navigators 'Location-felt', kun skriver javascript:, vil man få opp en kommandotolk, hvor man fortløpende kan taste inn kommandoer og se resutlatet umiddelbart.
- Eksempler:
- <A HREF="javascript: alert(Date())">nå</a> <FORM ACTION="javascript: '<H1><CENTE
og hopp</CENTER></H1>"'>

### 1.7.3 Programutførelsen

- Kode som står mellom <SCRIPT>- og </SCRIPT>-tagger, blir utført i den rekkefølge de fremkommer i html-dokumentet.
- Utførelsen skjer mens nettleseren analyserer dokumentet som lastes ned. Tidskrevende kode bør derfor utføres via en hendelseshåndterer, (eller i bakgrunnen, v.h.a. metoden setTimeout() ) slik at det ikke tar for lang tid for siden å vises.
- Kode som utføres kan referere til elementer som ikke er definert ennå.
- Funksjonsdefinisjoner utføres ikke før de kalles opp, og kan derfor referere til elementer som ennå ikke er definert.
- Hendelseshåndterere vet man ikke når utføres. Man må derfor passe på at elementene de refererer til allerede er definert. Det samme gjelder 'javascript:-URL'er' i hypertekst-lenker eller som verdi på ACTION-attributtet til html-skjema,
- Ved å definere alle funksjoner i html-hodet, er man garantert at alle funksjonene er definert før alle hendelseshåndtererne.
- Ved å sammenligne et element med null, kan man avgjøre om elementet er lastet eller ikke. Eksempel:

- if (parent.frames[1]==null) alert(”Rammen er ikke definert ennå. Prøv igjen om - litt”);
- Hendelseshåndtereren onLoad, som er attributt til <BODY> og <FRAMESET>, utføres når dokumentet
- er ferdig lastet. Ved å sette en gobal variabel til en bestemt verdi i denne, kan man ved å
- teste på denne variabelen avgjøre om dokumentet er ferdig lastet. Eksempel:
- <BODY onLoad=”status='ferdig”'> <A HREF=”javascript: if (status=='ferdig')alert('ferdig'); else alert ('ikke ferdig - ')”> </A>
- Hendelseshåndtereren onUnload, utføres et før et nytt dokument lastes.
- Et vindu-objekt eksisterer så lenge vinduet eller rammen det representerer eksisterer, men alle
- brukerdefinerte egenskaper ved vinduet, blir slettet når et nytt dokument lastes.

## 1.8 Vinduer og rammer

### 1.8.1 Vindusobjektets egenskaper.

- closed Boolsk verdi som viser om vinduet er stengt. F.o.m. JavaScript 1.1 default-Status For å vise eller sette tekststrengen som skal være 'default-verdien' til status-linjen.
- document Referanse til 'document'-objekt.
- frames[]|| Tabell over rammene i vinduet/rammen.
- history Referanse til 'history'-objekt.
- length Antall tilhørende rammer. Samme som frames.length
- location Referanse til 'location'-objekt.
- Math Objekt med diverse matematiske funksjoner og konstanter.
- name Vinduets/rammens navn
- navigator Referanse til 'navigator'-objekt.
- offscreenBuffering -
- opener Referanse til 'foreldre-vindu'. F.o.m. Javascript 1.1
- parent Referanse til 'foreldre-ramme'.
- screen Referanse til 'screen'-objekt.
- self Selvreferanse
- status For å vise eller sette tekststreng som vises i status-linjen.
- top Referanse til den 'rot-rammen'.
- window Selvreferanse

### 1.8.2 Vindusobjektets metoder

- alert() Dialogboks som viser:
- Tekst
- OK-knapp
- blur() -
- clearInterval() -
- clearTimeout() -
- close() Stenger vinduet/rammen.
- confirm() Dialogboks som viser:
- Tekst
- OK-knapp
- CANCEL-knapp
- focus() -
- moveBy() -
- moveTo() -
- open() -
- prompt() Dialogboks som viser:
- Tekst
- Tekstfelt
- OK-knapp
- CANCEL-knapp
- resizeBy() -
- resizeTo() -
- scroll() -
- scrollBy() -
- scrollTo() -
- setInterval() -
- setTimeout() -

### 1.8.3 Hendelseshånderere

- onblur
- ondragdrop
- onerror
- onfocus
- onload
- onmove
- onresize
- onunload

## 1.9 Dokumentobjektet

### 1.9.1 Dokumentobjektets egenskaper

- alinkColor Fargen på aktivisert hyperlenke. Kan settes i html-hode (eller <BODY ALINK=...>)
- anchors[]|| Tabell over ankerobjektene.
- applets[]|| Tabell over applet'ene. F.o.m. Javascript 1.1
- bgColor Bakgrunnsfargen.
- cookie Verdien av en 'cookie' assosiert med dokumentet.
- domain Angivelse av domene som kan gis høyere 'sikkerhetsklarering'. F.o.m. Javascript 1.1
- embeds[]|| Tabell over inbakte (embedded) objekter. F.o.m. Javascript 1.1
- fgColor Farge på teksten. Kan settes i html-hode (eller <BODY TEXT=...>)
- forms[]|| Tabell over dokumentets skjema.
- images[]|| Tabell over dokumentets bilder. F.o.m. Javascript 1.1
- lastModified Dato for siste endring.
- linkColor Fargen på hyperlenke som ikke er fulgt ennå. Kan settes i html-hode (eller <BODY LINK=...>)
- links[]|| Tabell over hyperlenkene.
- location Tekststreng som inneholder URL'en til dokumentet.
- plugins[]|| Samme som embeds[]||. F.o.m. Javascript 1.1
- referrer Tekststreng som inneholder URL'en til dokumentet som inneholdt hyperlenken som refererte til dokumentet.
- title Dokumentets tittel.
- URL Samme som location.
- vlinkColor Fargen på hyperlenke som er fulgt tiligere. Kan settes i html-hode (eller <BODY VLINK=...>)

### 1.9.2 'Document'-objektets metoder

- clear()
- close()
- open()
- write()
- writeln()

## 11.5 Øvelser

### 11.1

- Lag en android-applikasjon som gir brukeren en html-side som fungerer som et kommandolinjebasert "shell". Kommandolinjene skrives inn i et tekstfelt. Ved trykk på en knapp utføres kommandoen, og utskriften kommer til syne i html-siden sammen med tekstfeltet og knappen. Under finner du et kode-eksempel som demostrerer hvordan en prosess startes.

#### Oppstart av prosess

Listing 11.11: eksempler/11/AndroidLoader/src/no/hive/oopva60/loader/Loader.java

```
1  package no.hive.oopva60.loader;
2
3  public class Loader extends android.app.Activity {
4
5    java.util.ArrayList<java.lang.String> cmd;
6    android.webkit.WebView view;
7      java.lang.String str,typ;
8
9    @Override
10     public void onCreate(android.os.Bundle savedInstanceState) {
11
12     super.onCreate(savedInstanceState);
13         kjor("ls -l");
14     }
15
16   private void kjor(java.lang.String commandline) {
17
18     view = new android.webkit.WebView(this);
19     cmd = new java.util.ArrayList<String>();
20         for (String a: commandline.split(" "))
21           cmd.add(a);
22
23     java.lang.ProcessBuilder pb = new java.lang.ProcessBuilder(cmd);
24         typ="text/html";
25         setContentView(view);
26
27         try {
28       java.lang.Process p = pb.start();
29       java.util.Scanner pout = new java.util.Scanner(p.getInputStream());
```

```
30          str="";
31          while(pout.hasNextLine())
32                  str=str+pout.nextLine()+"<br>";
33
34      } catch (Exception e) {
35          str = e.getMessage();
36          typ = "text/plain";    }
37
38      view.loadData(str, typ, "utf-8");
39    }
40
41    @Override
42    public void onBackPressed() {
43
44      java.lang.System.exit(0);
45    }
46  }
```

# Kapittel 12

# CSS, Android-service og RMI

## 12.1 CSS

- CSS er en utvidelse av html.
- Brukes for å angi utseende.

### Syntax

selector { navn: verdi; navn: verdi; ... ; navn: verdi }

### Klasser

- Elementer kan grupperes ved å tilordne dem til klasser.

.sitat { color: green }
.hypotese {color: blue }
...
 dette blir et blått avsnitt 
 dette blir et grønt avsnitt 

#### Noen pseudoklasser

- link
- visited
- active
- first-line
- first-letter

### Id'er

- Enkelt-elementer kan gis identifikatorer

#viktigste { color: red }
...
 dette blir et rødt avsnitt 

### Fire måter å plassere stilsett i html-dokument

#### 1. Direkte i html-kode

<head>
<style media=screen> ....</style>
</head>

- (media kan også ha andre verdier. F.eks. print, projection, braille, ... )

#### 2. Lenke til eget stilsett-dokument

<head>
<link rel="stylesheet" type="text/css" href="..." />
</head>

#### 3. Importere stilsett i html-kode

<head>
<style type="text/css">
@import url(http://...)
i {color: red}
</head>

#### 4. Stilregler som attributter til html-elementer.

- selector trengs ikke – er underforstått

 Dette blir kursiv og rødt 

## 12.2 Android-Service

- En 'Service' er en komponent uten brukergrensesnitt, som tilbyr tjenester til andre komponeneter.

### Pensum til android-delen av denne forelesningen

`http://developer.android.com/guide/topics/fundamentals/services.html`

### To varianter

#### Startet

Service startes og kjører til den stoppes

#### Bundet

Service starter når en komponent binder seg til servicen. Den stopper når ingen komponenter lenger er bundet til den.

#### Livssyklus

`http://developer.android.com/images/service_lifecycle.png`

### Foreground

- Hvis brukeren skal oppfatte tjenesten.
- Ikke blant de første som ryddes av veien ved behov for minne.

### Notifications

- Toast
- Statusfelt

### To klasser å velge mellom

#### extends android.app.Service

- må sørge for at den stoppes
- må implementere onBind(). Hvis ikke binding skal tillates, kan metoden være 'return null;'
- må starte ny tråd fordi Service'en kjører i applikasjonens hovedtråd.

#### extends android.app.IntentService

- Lager egen arbeidstråd for å håndtere alle "intents" som kommer en og en gjennom en kø.
- Stopper tjenesten når alle forespørslene er behandlet.
- Gir default implementasjon av onBind() og onStart().
- Det eneste som gjenstår er å implementere onHandleIntent()

## 12.3 Eksempel Android-app med Service og CSS

### Service

Listing 12.1: eksempler/12/ServiceTest/src/ofa/servicetest/Mp3Service.java

```
1 package ofa.servicetest;
2
3 public class Mp3Service extends android.app.Service {
4
5   private android.media.MediaPlayer mp;
6
7   @Override
8   public void onStart(android.content.Intent intent, int startId) {
```

```
9
10      super.onStart(intent, startId);
11      mp = android.media.MediaPlayer.create(this, R.raw.forelesning);
12      mp.start();
13      android.widget.Toast.makeText(
14          this,"Mp3Service er startet", android.widget.Toast.LENGTH_SHORT).
                show();
15    }
16
17    @Override
18    public void onDestroy() {
19
20      super.onDestroy();
21      android.widget.Toast.makeText(
22          this,"Mp3Service er avsluttet", android.widget.Toast.LENGTH_SHORT).
                show();
23      mp.release();
24    }
25
26    @Override
27    public android.os.IBinder onBind(android.content.Intent intent) {
28      return null;
29    }
30
31  }
```

Listing 12.2: eksempler/12/ServiceTest/src/ofa/servicetest/ServiceTest.java

```
1  package ofa.servicetest;
2
3  public class ServiceTest extends android.app.Activity {
4
5    private android.content.Intent intent;
6    private android.webkit.WebView wv;
7
8      @Override
9      public void onCreate(android.os.Bundle savedInstanceState) {
10
11       super.onCreate(savedInstanceState);
12
13         intent = new android.content.Intent(this, Mp3Service.class);
14
15         wv = new android.webkit.WebView(this);
16
17         wv.loadUrl("file:///android_asset/index.html");
18         wv.getSettings().setJavaScriptEnabled(true);
19         wv.addJavascriptInterface(
20             new JavaScriptGrensesnitt(this),
21             "javaobjekt"
22         );
23
24         setContentView(wv);
25     }
26
27   public class JavaScriptGrensesnitt {
28
29     android.content.Context kontekst;
30
31     public JavaScriptGrensesnitt(android.content.Context kontekst) {
```

```
32        super();
33      }
34
35      public void startMp3(){
36        startService(intent);
37      }
38
39      public void stoppMp3() {
40        stopService(intent);
41      }
42
43    }
44  }
```

Listing 12.3: eksempler/12/ServiceTest/AndroidManifest.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3        package="ofa.servicetest"
4        android:versionCode="1"
5        android:versionName="1.0">
6      <application android:icon="@drawable/icon" android:label="@string/
           app_name">
7          <activity android:name=".ServiceTest"
8                    android:label="@string/app_name">
9              <intent-filter>
10                 <action android:name="android.intent.action.MAIN" />
11                 <category android:name="android.intent.category.LAUNCHER" /
                       >
12             </intent-filter>
13         </activity>
14
15         <service android:name=".Mp3Service"/>
16
17     </application>
18     <uses-sdk android:minSdkVersion="8" />
19
20 </manifest>
```

## CSS

Listing 12.4: eksempler/12/ServiceTest/assets/index.html

```
1  <?xml version="1.0" encoding="UTF-8"?>
2
3  <!DOCTYPE
4      html PUBLIC "-//W3C//DTD XHTML Basic 1.1//EN"
5      "http://www.w3.org/TR/xhtml-basic/xhtml-basic11.dtd"
6      >
7
8  <html>
9
10   <head>
11     <meta name='viewport' content='width=device-width, user-scalable=no'/>
12     <link rel='stylesheet' href='mobil.css' type='text/css' media='screen
           and (max-width: 480px)' />
13     <title>ServiceTest</title>
14   </head>
15
```

```
16    <body>
17    <ul>
18
19        <li onclick="javaobjekt.startMp3()">
20            Start
21        </li>
22
23        <li onclick="javaobjekt.stoppMp3()">
24           Stopp
25        </button>
26
27    </ul>
28    </body>
29
30 </html>
```

Listing 12.5: eksempler/12/ServiceTest/assets/mobil.css

```
1  body {
2
3      background-color: black;
4      font-family: Arial;
5  }
6
7  ul {
8
9      list-style: none;
10     margin: 50px;
11     padding: 0;
12 }
13
14 ul li  {
15
16     display: block;
17     color: black;
18     font-size: 40px;
19     font-weight: bold;
20 }
21
22 ul li:first-child  {
23
24     -webkit-border-top-left-radius: 8px;
25     -webkit-border-top-right-radius: 8px;
26     background-color: green;
27 }
28
29 ul li:last-child  {
30     background-color: red;
31     -webkit-border-bottom-left-radius: 8px;
32     -webkit-border-bottom-right-radius: 8px;
33 }
```

## 12.4 RMI

### Tradisjonelt to kommunikasjonsformer:

- overføring
- "fjernstyring"

**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.

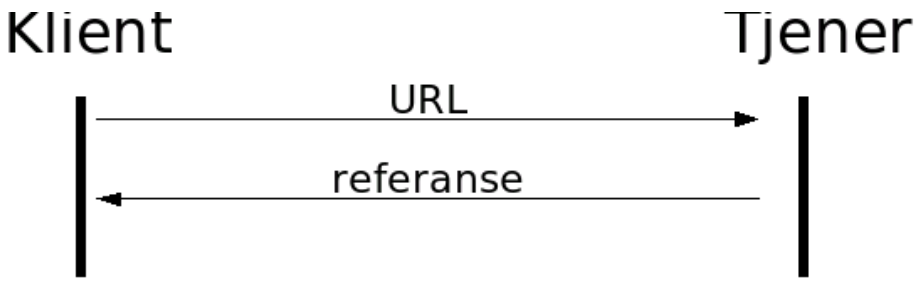
**2.**

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

## 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.

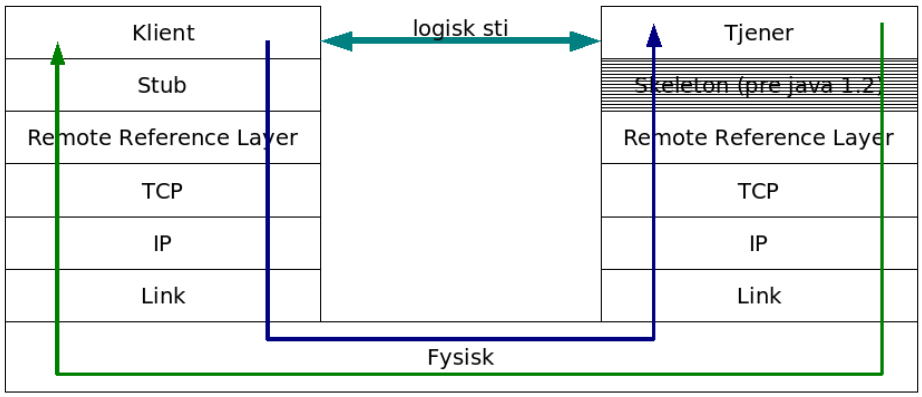
**figur**




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).

## 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.

Listing 12.6: 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.

Listing 12.7: 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 }
```

### 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.

Listing 12.8: 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 "RemoteExeption" 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)

Listing 12.9: 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 }
```

## 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

Listing 12.10: eksempler/Hallogrensesnitt.java

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

Listing 12.11: 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 }
```

Listing 12.12: 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=klien*

Listing 12.13: 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 }
```

Listing 12.14: 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=tjener.po*

Listing 12.15: 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 }
```

Listing 12.16: eksempler/tjener.policy

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

## 12.5 Øvelser

### 12.1

Utvid Service-eksemplet, slik at det blir mulig å sette avspillingen på pause og fortsette avspillingen etterpå.

### 12.2

- 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)

- Gjør klientprogrammet web-basert.
- Velg CGI, servlet eller JSP.
- Gi en faglig begrunnelse for valget.

#### d)

Lag to ulike stilsett: Ett for PC ett for telefon.

#### e)

Sørg for at Klientprogrammet automatisk bruker riktig stilsett.

### 12.3

Gjør om din løsning på oppgave 12.2, 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.

# Kapittel 13

# Oppsummering del 2

## 13.1 Løsningsforslag

### 12.2.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).

### 12.2.b)

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

Listing 13.1: losninger/12/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 }
```

Listing 13.2: losninger/12/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  }
```

Listing 13.3: losninger/12/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 }
```

Listing 13.4: losninger/12/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  }
```

Listing 13.5: losninger/12/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  }
```

## 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.

- Kjøringen trigges av http-forespørsel fra web-klient.
- Produserer dokument (f.eks. et html-dokument) som returneres til klienten.
- Servlets-container laster, initialiserer og kjører servlets.

### Eksempler

#### Eksempel 2: Input fra html-skjema

Listing 13.6: eksempler/08/skjemaservlet/skjemaservlet.html

```
<html>
<form action="SkjemaServlet">
Brukernavn: <input type=text name=Brukernavn> <p>
<input type=submit>
</form>
</html>
```

Listing 13.7: eksempler/08/SkjemaServlet.java

```
public class SkjemaServlet  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+"!");

        ut.flush();
    }
}
```

### Eksempel 3: Informasjonskapsler

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

Listing 13.8: eksempler/08/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,
               javax.servlet.ServletException {

      java.io.PrintWriter ut;
      javax.servlet.http.Cookie nybaktKake;
      javax.servlet.http.Cookie[] kake;
      int i;

      ut = respons.getWriter();
      respons.setContentType("text/html");

      kake = request.getCookies();
      if (kake == null || kake.length == 0) {

      nybaktKake = new javax.servlet.http.Cookie("IP", request.
          getRemoteAddr() );
      nybaktKake.setMaxAge(60*60);
      respons.addCookie(nybaktKake);
```

```
24        ut.println("Velkommen fremmede");
25
26        } else {
27        // skriver ut verdiene i informasjonskapslene
28        ut.printf("Takk for sist! Du har %d kake(r)<p>", kake.length );
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

Listing 13.9: eksempler/08/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 }
```

## 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:**

Listing 13.10: eksempler/09/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 enJSP-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)

**Et eksempel med kommentarer, direktiver og uttrykk:**

Listing 13.11: eksempler/09/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:**

Listing 13.12: eksempler/09/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

**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 implementer Serializable
- pakk eventuelt inn i jar-fil sammen med manifest-fil (påkrevd for bruk i IDE)

## 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

- Eksemplet under består av en java-fil og en html-fil. Det viser tre måter å kopiere verdier fra html-skjema til en bønne, UTEN å bruke 'value'-attributtet.

Listing 13.13: eksempler/10/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 }
```

Listing 13.14: eksempler/10/person1.jsp

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

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

<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>
```

## 13.5 JavaScript

- Standard (ECMA) skriptspråk som er implementert i de mest vanlige nettleserne.
- Under ser du noen eksempler, hvor javascript er innfelt blant html-koden. Skriptene blir utført på klienten (nettleseren).

Listing 13.15: eksempler/11/javascripthallo.html

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

```
<html>

  <head>
    <meta name='viewport'
    content='width=device-width, user-scalable=yes, initial-scale=1.2'
    />
    <meta http-equiv="Content-type" content="text/html;charset=UTF-8">
    <title>JavaScript-hallo</title>
  </head>

  <body>

    <script type="text/javascript">
      document.write("<h1 id='hilsen'>Hallo</h1>");
      overskrift = document.getElementById('hilsen');
    </script>

    <button onclick="overskrift.innerHTML='Borte'">
      Klikk her
    </button>

  </body>

</html>
```

Listing 13.16: eksempler/11/rullende_ramme.html

```
<html>
<SCRIPT>
function blabla ()
{
  var rammenummer;
  for (rammenummer=0; rammenummer<window.length;rammenummer++)
  {
      window.frames[rammenummer].document.open("text/html");

      for (var teller=0; teller < 1500; teller++)
              window.frames[rammenummer].document.write("bla ");

      window.frames[rammenummer].document.close();
      window.frames[rammenummer].posisjon=0;
      window.frames[rammenummer].offset=rammenummer+1;
      setInterval("rull(window.frames["+rammenummer+"])",10);

  }
}

function rull(ramme)
{

  if (ramme.posisjon > ramme.innerHeight*2 || ramme.posisjon < 0)
    ramme.offset*=-1;
  ramme.scrollBy(0,ramme.offset);
  ramme.posisjon+=ramme.offset;
}
</SCRIPT>
<FRAMESET rows=20%,20% cols=50%,50% onLoad=blabla()>
<FRAME NAME=ramme1 SRC="javascript:'<h1>Ramme1</h2>'">
<FRAME NAME=ramme2 SRC="javascript:'<h1>Ramme2</h2>'">
```

```
33 <FRAME NAME=ramme3 SRC="javascript:'<h1>Ramme3</h2>'">
34 <FRAME NAME=ramme4 SRC="javascript:'<h1>Ramme4</h2>'">
35 </FRAMESET>
36
37 </hmtl>
```

## 13.6 CSS

- CSS er en utvidelse av html.
- Brukes for å angi utseende.

### Syntax

selector { navn: verdi; navn: verdi; ... ; navn: verdi }

### Klasser

- Elementer kan grupperes ved å tilordne dem til klasser.

.sitat { color: green }
.hypotese {color: blue }
...
 dette blir et blått avsnitt 
 dette blir et grønt avsnitt 

### Id'er

- Enkelt-elementer kan gis identifikatorer

#viktigste { color: red }
...
 dette blir et rødt avsnitt 

### Fire måter å plassere stilsett i html-dokument

#### 1. Direkte i html-kode

<head>
<style media=screen> ....</style>
</head>

- (media kan også ha andre verdier. F.eks. print, projection, braille, ... )

#### 2. Lenke til eget stilsett-dokument

<head>
<link rel="stylesheet" type="text/css" href="..." />
</head>

**3. Importere stilsett i html-kode**

<head>
<style type=”text/css”>
@import url(http://...)
i {color: red}
</head>

**4. Stilregler som attributter til html-elementer.**

- selector trengs ikke – er underforstått

<i style=”color: red”> Dette blir kursiv og rødt </i>

## Eksempel

Listing 13.17: eksempler/12/ServiceTest/assets/index.html

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE
    html PUBLIC "-//W3C//DTD XHTML Basic 1.1//EN"
    "http://www.w3.org/TR/xhtml-basic/xhtml-basic11.dtd"
    >

<html>

  <head>
    <meta name='viewport' content='width=device-width, user-scalable=no'/>
    <link rel='stylesheet' href='mobil.css' type='text/css' media='screen
        and (max-width: 480px)' />
    <title>ServiceTest</title>
  </head>

  <body>
  <ul>

      <li onclick="javaobjekt.startMp3()">
          Start
      </li>

      <li onclick="javaobjekt.stoppMp3()">
        Stopp
      </button>

  </ul>
  </body>

</html>
```

Listing 13.18: eksempler/12/ServiceTest/assets/mobil.css

```
body {

    background-color: black;
    font-family: Arial;
}
```

```
6
7  ul {
8
9      list-style: none;
10     margin: 50px;
11     padding: 0;
12 }
13
14 ul li  {
15
16     display: block;
17     color: black;
18     font-size: 40px;
19     font-weight: bold;
20 }
21
22 ul li:first-child  {
23
24     -webkit-border-top-left-radius: 8px;
25     -webkit-border-top-right-radius: 8px;
26     background-color: green;
27 }
28
29 ul li:last-child  {
30     background-color: red;
31     -webkit-border-bottom-left-radius: 8px;
32     -webkit-border-bottom-right-radius: 8px;
33 }
```

## 13.7 Android

### Android-Service

- En 'Service' er en komponent uten brukergrensesnitt, som tilbyr tjenester til andre komponeneter.

#### To varianter

**Startet** Service startes og kjører til den stoppes

**Bundet** Service starter når en komponent binder seg til servicen. Den stopper når ingen komponenter lenger er bundet til den.

**Livssyklus** `http://developer.android.com/images/service_lifecycle.png`

#### Foreground

- Hvis brukeren skal oppfatte tjenesten.
- Ikke blant de første som ryddes av veien ved behov for minne.

#### Notifications

- Toast
- Statusfelt

**To klasser å velge mellom**

**extends android.app.Service**

- må sørge for at den stoppes
- må implementere onBind(). Hvis ikke binding skal tillates, kan metoden være 'return null;'
- må starte ny tråd fordi Service'en kjører i applikasjonens hovedtråd.

**extends android.app.IntentService**

- Lager egen arbeidstråd for å håndtere alle "intents" som kommer en og en gjennom en kø.
- Stopper tjenesten når alle forespørslene er behandlet.
- Gir default implementasjon av onBind() og onStart().
- Det eneste som gjenstår er å implementere onHandleIntent()

## Eksempel

- Eksemplet demonstrer brukh av service og hvordan javascript kan kobles til java-objekter.

Listing 13.19: eksempler/12/ServiceTest/src/ofa/servicetest/Mp3Service.java

```
1  package ofa.servicetest;
2
3  public class Mp3Service extends android.app.Service {
4
5    private android.media.MediaPlayer mp;
6
7    @Override
8    public void onStart(android.content.Intent intent, int startId) {
9
10     super.onStart(intent, startId);
11     mp = android.media.MediaPlayer.create(this, R.raw.forelesning);
12     mp.start();
13     android.widget.Toast.makeText(
14         this,"Mp3Service er startet", android.widget.Toast.LENGTH_SHORT).
               show();
15   }
16
17   @Override
18   public void onDestroy() {
19
20     super.onDestroy();
21     android.widget.Toast.makeText(
22         this,"Mp3Service er avsluttet", android.widget.Toast.LENGTH_SHORT).
               show();
23     mp.release();
24   }
```

```
25
26    @Override
27    public android.os.IBinder onBind(android.content.Intent intent) {
28      return null;
29    }
30
31  }
```

Listing 13.20: eksempler/12/ServiceTest/src/ofa/servicetest/ServiceTest.java

```
1  package ofa.servicetest;
2
3  public class ServiceTest extends android.app.Activity {
4
5    private android.content.Intent intent;
6    private android.webkit.WebView wv;
7
8      @Override
9      public void onCreate(android.os.Bundle savedInstanceState) {
10
11       super.onCreate(savedInstanceState);
12
13         intent = new android.content.Intent(this, Mp3Service.class);
14
15         wv = new android.webkit.WebView(this);
16
17         wv.loadUrl("file:///android_asset/index.html");
18         wv.getSettings().setJavaScriptEnabled(true);
19         wv.addJavascriptInterface(
20              new JavaScriptGrensesnitt(this),
21              "javaobjekt"
22         );
23
24         setContentView(wv);
25     }
26
27   public class JavaScriptGrensesnitt {
28
29     android.content.Context kontekst;
30
31     public JavaScriptGrensesnitt(android.content.Context kontekst) {
32       super();
33     }
34
35     public void startMp3(){
36       startService(intent);
37     }
38
39     public void stoppMp3() {
40       stopService(intent);
41     }
42
43   }
44 }
```

Listing 13.21: eksempler/12/ServiceTest/AndroidManifest.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
```

```
3         package="ofa.servicetest"
4         android:versionCode="1"
5         android:versionName="1.0">
6     <application android:icon="@drawable/icon" android:label="@string/
          app_name">
7         <activity android:name=".ServiceTest"
8                   android:label="@string/app_name">
9             <intent-filter>
10                <action android:name="android.intent.action.MAIN" />
11                <category android:name="android.intent.category.LAUNCHER" /
                      >
12            </intent-filter>
13        </activity>
14
15        <service android:name=".Mp3Service"/>
16
17    </application>
18    <uses-sdk android:minSdkVersion="8" />
19
20 </manifest>
```

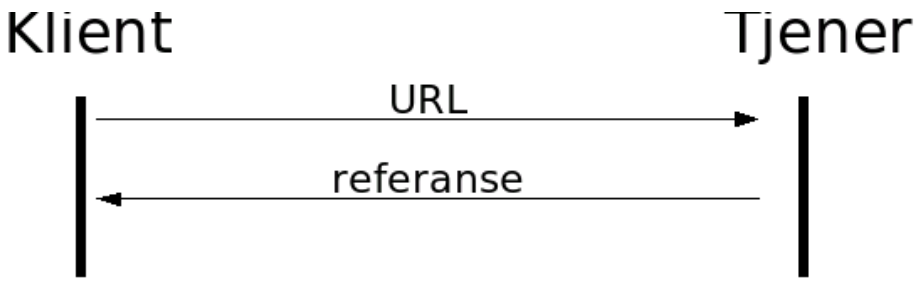
## 13.8 RMI

- RMI er forkortelse for 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.

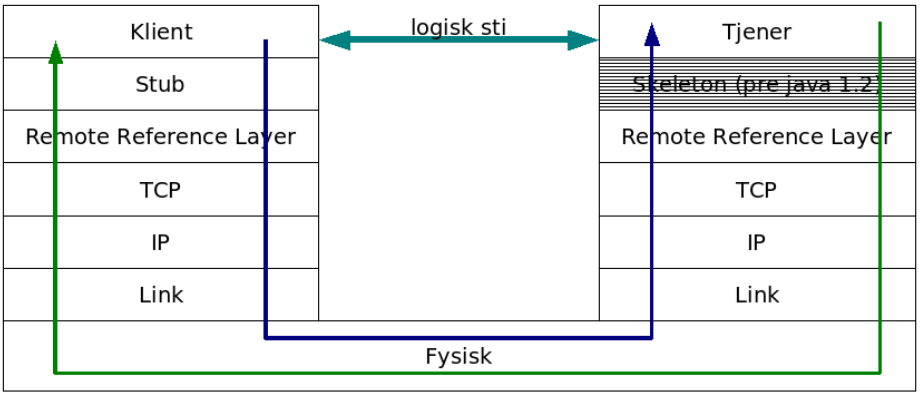
### figur




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




## Eksempel

### Tjener

Listing 13.22: eksempler/Hallogrensesnitt.java

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

Listing 13.23: 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 }
```

Listing 13.24: 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  }
```

Listing 13.25: eksempler/tjener.policy

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

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

**Klient**

Listing 13.26: 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 }
```

Listing 13.27: eksempler/klient.policy

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

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

## 13.9 Eksamensoppgave-eksempler

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

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

Både A og B har metoden getC. Beskriv kort hovedforskjellene mellom å kjøre *a.getC()* og å *b.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 et javascript, android applikasjon, servlet, JSP eller bønne (bean). Begrunn svaret. ... ... ...

# Del III

# Vedlegg

# Tillegg A

# Eksamen våren 2010 - med løsningsforslag

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.

## Oppgave 1 (20%)

Applet, servlet, JSP eller bønne (bean)? Disse teknologiene har ulike egenskaper.

### a)

Gi et eksempel på et program du ville implementert som *applet*. Begrunn svaret.

#### svar:

En videokonferanse-klient, som skal være tilgjengelig på en nettside setter opp socket-forbindelser til en sentral tjener som administrerer konferansen og videreformidler lyd og bilde.

Siden den skal den både må ha tilgang til lokale resursser som mikrofon og kamera, må klienten kjøre lokalt. Verken servlet, JSP vil dermed være aktuelt. Bønne vil eventuelt kun være aktuelt som en del av applet'en.

### b)

Gi et eksempel på et program du ville implementert som *servlet*. Begrunn svaret.

#### svar:

Et program tar i mot forespørsler om reservering av hotellrom, logger reservasjonen i en fil-tjener og sender forespørslen via e-post til hotellet. Web-klienten skal kun få en enkel kvittering tilbake.

Siden den ikke skal bruke lokale resursser, er det unødvendig å bruke applet. Applets har lang oppstartstid.

Siden applikasjonene er så begrenset, velger jeg å ikke dele den opp i komponenter. Det blir da mer java-kode enn html-kode i applikajsonen. Det blir dermed mest ryddig å skrive den som en servlet fremfor en JSP. Den skal ta imot data fra en web-klient og sende tilbake, og kan dermed ikke som helhet være en bønne.

### c)

Gi et eksempel på et program du ville implementert som *JSP*. Begrunn svaret.

**svar:**

Et program som produserer web-side for reservasjon av hotellrom.

Siden den ikke skal bruke lokale resursser, er det unødvendig å bruke applet. Applets har lang oppstartstid.

En vesentlig del av koden vil være utskrift av html-kode. Dette taler mot servlet, og til fordel for JSP.

Den skal ta imot data fra en web-klient og sende tilbake, og kan dermed ikke som helhet være en bønne.

### d)

Gi et eksempel på et program du ville implementert som *bønne*. Begrunn svaret.

**svar:**

Et program som vedlikeholder en sentral database over tilgjengelig hotellrom for en mengde hoteller, med funksjoner for å reservere og frigjøre rom. Programmet skal brukes i JSP-en fra c) og i programvare for hotell-resepsjonistene.

Siden funksjonaliteten skal gjenbrukes i ulike sammenhenger, vil en bønne være tingen. Den kan brukes i vanlige java programmer og er godt støttet i JSP.

## Oppgave 2 (20%)

Skriv et java-program som sender et datagram v.h.a. transportprotokollen UDP. Inkluder forklarende kommentarer.

```
1  public class DatagramSender {
2    public static void main(String[] args) throws java.io.IOException {
3
4      // Oppretter socket
5      java.net.DatagramSocket s = new java.net.DatagramSocket();
6
7      // Oppretter datagram
8      java.net.DatagramPacket d = new java.net.DatagramPacket(
9      ``test''.getBytes(),// konstruerer byte-array av tekststrengen ``test''
10     ``test''.length(), // lengden av byte-arrayet
11     java.net.InetAddress.getByName(// mottagerens ip-adresse
12     ``debbie.hive.no''),
13     8888// mottagerens port-nummer
14     );
15     s.send(d); // sender pakken d med socket'en s
```

```
16      s.close(); // stenger socket'en
17      }
18  }
```

# Oppgave 3 (20%)

To tråder, *t1* og *t2* skal oppdatere en variabel, v1. Forklar og vis med kode-utdrag hvordan vi, med Java, kan sørge for at det ikke oppstår en kappløpsituasjon (race condition).

### svar:

Ved å sørge for at variabelen er *private* i et objekt som har *synchroninzed* metoder for oppdatering, vil vi hindre kappløp.

Klassen som variabelen kan instasieres fra kan se slik ut:

```
1  class V {
2
3    private int value;
4
5    public synchronized void setV1(int value) { this.value = value; }
6    public synchronized int getV1() { return value; }
7  }
```

Instansieringen kan gjøres slik:

```
1  V v1 = new V();
```

Trådene t1 og t2 vil da måtte få en referanse til v1, f.eks. som paramenter i konstruktøren. Slik:

```
1  Traad t1 = new Traad(v1);
2  Traad t2 = new Traad(v1);
```

Dette vil sørge for at kun en av trådene får tilgang til v1 om gangen. Dermed er ikke kappløp-situasjoner mulig.

# Oppgave 4 (10%)

Et web-basert program tar i mot bestillinger fra følgende skjema:

```
1  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
2  <html><head><title>Eksempel</title></head>
3    <body>
4      <form action=eksempel>
5        <input type=text name=varenavn>
6        <input type=text name=pris>
7        <input type=text name=antall>
8        <input type=submit>
9      </form>
10   </body>
11 </html>
```

Gi en beskrivelse av dette skjemaet.

**svar:**

Skjemaet befinner seg i et html-dokument med tittelen *Eksempel*. Det består av skjema-elementer; tre tekstfelt og et send-knapp. De tre tekstfeltene har navnene varepris, pris og antall. Skjemaet har også attributtet *action*, som er en relativ url som skjemaet skal sendes til.

Hvis f.eks. url'en til dokumentet var *http://debbie.hive.no/skjema.html*, og skjemaet var fyllt ut med 'X' i varenavn '10' i varepris '50' i pris og '5' i antall, så ville et klikk på submit-knappen føre til en http-get med mot url'en

*http://debbie.hive.no/eksempel?varenavn=X&varepris=5&pris=10&antall=2.*

# Oppgave 5 (30%)

Programmet nevnt i Oppgave 4, sender en kvittering tilbake til klienten. Kvitteringen er på samme form som eksemplet under:

```
5 stk. brød av kr. 10.
Tilsammen 50 kr.
```

Hvordan ville du implementert dette programmet, som ...

## a)

et CGI-skript skrevet i Java. Vær konkret, skriv gjerne kode.

**svar:**

```
1
2  // Skriver ut http-hode for "plain-tekst", og en tom linje for angi slutt
       paa hodet.
3  System.out.println("Content-type:text/txt;charset=utf-8\n\n");
4
5  // Henter miljoevariablen QUERY_STRING som inneholder "skjemaet"
6  String[] sporring = System.getenv("QUERY_STRING").split("&");
7
8  // Putter sporring i en hashmap
9  Map<String, String> skjema = new HashMap<String, String>();
10 for (String variabel : sporring)
11   skjema.put( variabel.split("=")[0], variabel.split("=")[0] );
12
13 // Tilordner variabler
14 Integer antall  = Integer.parseInt(skjema.get("antall"));
15 String varenavn = skjema.get("varenavn");
16 Float pris      = Float.parseFloat(skjema.get("varepris"));
17 Float sum       = pris*antall;
18
19 // Skriver ut kvittering.
20 System.out.printf(
21   "%d stk. %s av kr. %f.\nTilsammen %f kr.\n",
22   antall,
23   varenavn,
24   pris,
25   sum
26 );
```

Et cgi-skript som er kjørbart som cgi-program for web-tjeneren må lages.

Hvis hovedklassen til programmet som koden over er hentet fra heter Bestillingsbehandling, så kan skriptet f.eks. se slik ut:

```
#!/bin/sh
java Bestillingsbehandling
```

## b)

en servlet. Vær konkret, skriv gjerne kode.

```
public class SkjemaServlet  extends HttpServlet {

    public void doGet( request, response )

        throws IOException, ServletException {

        response.setContentType(``text/Plain'');
        PrintWriter ut = respons.getWriter();

        // Tilordner variabler
        Integer antall   = Integer.parseInt(request.getParameter("antall"))
            ;
        String varenavn  = request.getParameter("varenavn");
        Float pris    = Float.parseFloat(request.getParameter("pris"));
        Float sum        = pris*antall;

        ut.printf(
          ``%d stk. %s av kr. %f.\nTilsammen %f kr.\n'',
          antall,
          varenavn,
          pris,
          sum
        );

        ut.flush();
    }
}
```

## c)

en JSP. Vær konkret, skriv gjerne kode.

### svar:

```

<%@ page contentType="text/plain" pageEncoding="UTF-8" %>
<%
  // Tilordner variabler

        Integer antall   = Integer.parseInt(request.getParameter("antall"))
            ;
        String varenavn  = request.getParameter("varenavn");
        Float pris    = Float.parseFloat(request.getParameter("pris"));
        Float sum        = pris*antall;
%>

```

```
12 <%= antall%> stk. <%= varenavn %> av kr. <%= pris %>
13 Tilsammen <%= sum %> kr.
```

# Tillegg B

# Eksamen våren 2009 - med løsningsforslag

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.

## 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.

### a)

Komponent som oppdaterer vare-lager-databasen

### 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.

### b)

Komponent som utfører penge-transaksjoner

### Løsningsforslag

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

### c)

En komponent som skriver logg til fil

### 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.

### d)

Brukergrensesnitt for å betale

### Løsningsforslag

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

### e)

Brukergrensesnitt for å handle

### 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.

## Oppgave 2 (25%)

### a)

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

### 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.

### b)

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

### 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.

## c)

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

### 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.

## d)

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

### Løsningsforslag

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

## e)

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

### 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 aa skrive til loggen
6     }
7 }
```

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

# 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  }
```

## a)

Gi en kort overordnet beskrivelse av hva koden gjør

## 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.

## b)

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

## Løsningsforslag

### 10

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

### 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'.

### 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

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.

**16**

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

**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()*.

**18**

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

**19**

Den indre løkka avsluttes.

**20**

Den ytre løkka avsluttes.

## c)

Hva blir utskriften av koden?

## Løsningsforslag

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

## d)

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

## 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.

### 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.

### Løsningsforslag

```
1  // Grensensittet som skal implementeres og gjoeres kjent for klientene
2  public interface Ekkogrensesnitt extends java.rmi.Remote{
3
4    public String ekko(String tekst) throws java.rmi.RemoteException;
5  }
6
7
8  // Implementasjon av grensesnittet
9  public class EkkoImpl
10 extends java.rmi.server.UnicastRemoteObject
11 implements Ekkogrensesnitt{
12
13   public EkkoImpl() throws java.rmi.RemoteException {
14
15   }
16
17   public String ekko(String tekst) throws java.rmi.RemoteException {
18
19     return tekst;
20   }
21 }
22
23 // Selve tjeneren starter navneregisteret og instansierer implementasjonen.
24 // Objektet (sammen med et navn)registreres i navneregisteret.
25 public class EkkoTjener {
26
27   public static void main (String[] args) throws
28   java.net.MalformedURLException,
29   java.rmi.RemoteException{
30
31     java.rmi.registry.LocateRegistry.createRegistry(1099);
32     java.rmi.Naming.rebind(
33         "rmi://vert.domene/Ekko",
34         new EkkoImpl()
35     );
36   }
37 }
```

# 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>
  </body>
</html>
```

## 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.

## 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.

## 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:

```
String queryString    = System.getenv("QUERY\_STRING")
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.

## c)

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

## Løsningsforslag

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

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

```

```
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  }
```

## d)

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

### 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.

## e)

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

### Løsningsforslag

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

# Tillegg C

# Applets

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

## C.1 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.

## C.2 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

## C.3 preSwing-applet

- extends java.applet.Applet

```
java.lang.Object|-- java.awt.Component|----java.awt.Container|------java.awt.Panel|---
```

### java.applet.Applet.paint()

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

### Eksempel:

Listing C.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 }
```

Listing C.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>
```

## C.4 Swing-applet

- extends javax.swing.JApplet

```
java.lang.Object|-- java.awt.Component|----java.awt.Container|------java.awt.Panel|---
```

- Unngår direkte opptegning (painting)

### javax.swing.JApplet.init()

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

### Eksempel:

Listing C.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 }
```

Listing C.4: eksempler/halloApplet2.html

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

## C.5 Eksempel på interaktiv applet

Listing C.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 }
```

Listing C.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>
```

## C.6 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

Listing C.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`

Listing C.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**

Listing C.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`

Listing C.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);
  }
}
```

Listing C.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`

Listing C.12: eksempler/BildeApplet3.java

```
public class BildeApplet3 extends javax.swing.JApplet {

  javax.swing.ImageIcon bilde;

  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()

## C.7 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

Listing C.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`

Listing C.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

Listing C.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

Listing C.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);
    panel.add(loop);

    this.add(panel);
  }

  public void actionPerformed(
      java.awt.event.ActionEvent e) {

    Object o = e.getSource();

```

```
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  }
```