Thomas Nordli

Notater til OOPVA60

3. utgave

Horten, Høgskolen i Vestfold, 2013

Kompendium 2/2013

Kompendium 2/2013 Høgskolen i Vestfold

ISSN 0808-1328

Innhold

Om skriftet

Skriftet *Notater til OOPVA60, tredje utgave* er et kompendium som inneholder undervisningsmateriale. Undervisningsmaterialet er produsert i forbindelse med undervisning i faget *Objektorientert programmering II.* , f.o.m våren 2009 t.o.m. våren 2013. 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.

Om forfatteren

Forfatteren, Thomas Nordli, har vært fagansvarlig for kurset. Han er ansatt som høgskolelektor ved *Fakultet for teknologi og maritime fag* ved *Høgskolen i Vestfold*, hvor han underviser innen emnene *operativsystemer*, *databaser*, *nettverk*, *kybersikkerhet* og *programmering.* I tillegg er han lærer og kurshefte-forfatter for NKI-kurset *Operativsystemet Linux.*

Endringer fra første til andre 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.

Endringer fra andre til tredje utgave

Foruten små forbedringer her og der, inneholder denne utgaven flere nye oppgaver (både øvinger, oblig'er og eksamensoppgaver).

Vedlegget om applets er utelatt i denne utgaven.

Del I

TCP/IP

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"

1.1.1 lagdelt modell

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

1.1.2 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.

1.1.3 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

1.2.1 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.

1.4.1 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)  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
22                  utskriver.format(
23                      "Interessant at du sier \"%s\".\n",
24                      innleser.nextLine() );
25                  utskriver.flush();
26              }
27              forbindelse.close();
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();

1.4.2 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

1.5.1 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.

1.5.2 Oppgave 1.2

Programmer en klient til eksempelprogrammet TCPSkravler_1.

1.5.3 Oppgave 1.3

Programmer en klient til eksempelprogrammet UDPSensor.

Kapittel 2

Threads, synkronisering

2.1 Løsningsforslag på forrige ukes øving

2.1.1 Oppgave 1.2

- Programmer en klient til eksempelprogrammet TCPSkravler_1.

Listing 2.1: 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 }
```

2.1.2 Oppgave 1.3

Programmer en klient til eksempelprogrammet UDPSensor

Listing 2.2: 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. lage klasse som implementerer grensesnittet (interface) Runnable

2.2.1 Eksempler på bruk av tråder i java

extends Thread

Listing 2.3: eksempler/02/Traad_1.java

```
1
2  public class Traad_1 extends Thread {
3
4      /*
5       * Eksempel paa traad som utvidelse av klassen Thread
6       */
7
8      public static void main(String[] args) throws
9          java.io.IOException {
10
11         // Deklarerer
12         Traad_1 t1, t2;
13
14         // Instansierer
15         t1 = new Traad_1();
16         t2 = new Traad_1();
17
18         // Starter
19         t1.start();
20         t2.start();
21     }
22
23
24     public void run() { // kjoeres naar traaden startes
25
26         for (int i=0; i<5; i++)
27             System.out.println(this.getName());
28     }
29 }
```

Listing 2.4: eksempler/02/Traad_2.java

```
1  public class Traad_2 extends Thread {
2
3      /*
4       *  Demonsterer konkurrerende kjoering av fem traader
5       */
6
7
8      public static void main(String[] args) throws
9          java.io.IOException {
10
11         int i;
12         for (i=0; i<5; i++)
13             new Traad_2().start();
14     }
15
16     public void run() {
17
18         int i;
19         for (i=0; i<5; i++) {
20             System.out.println(this.getName());
21             try {
22                 sleep( (int)(Math.random()*1000) );
23
24             } catch (InterruptedException e) {
25                 e.printStackTrace();
26             }
```

```
27          }
28      }
29  }
```

implements Runnable

Listing 2.5: eksempler/02/Traad_3.java

```
1  /*
2   * Demonsterer traad som et objekt som implementerer grensesnittet Runnable
3   */
4
5  public class Traad_3 implements Runnable {
6
7      public static void main(String[] args) {
8
9          new Traad_3();
10     }
11
12     public Traad_3 () {
13         int i;
14         for (i=0; i<5; i++)
15             new Thread(this).start();
16     }
17
18     public void run() {
19
20         int i;
21         for (i=0; i<5; i++) {
22             System.out.println(Thread.currentThread().getName());
23             try {
24                 Thread.sleep( (int)(Math.random()*1000) );
25
26             } catch (InterruptedException e) {
27                 e.printStackTrace();
28             }
29         }
30     }
31 }
```

TCPSkravler

Listing 2.6: eksempler/02/TCPSkravler_2.java

```
1  import java.io.IOException;
2  import java.io.PrintWriter;
3  import java.net.ServerSocket;
4  import java.net.Socket;
5  import java.util.Scanner;
6
7
8  public class TCPSkravler_2 extends Thread {
9
10     /*
11      *  Demonstrer bruk av traader til klienthaandtering i tjener
12      */
13
14     public  Socket forbindelse;
15
```

```
16      public static void main(String[] args) throws IOException {
17
18          ServerSocket tjenerSocket = new ServerSocket(8888);
19          tjenerSocket.setReuseAddress(true);
20
21          while (true)
22              new TCPSkravler_2(tjenerSocket.accept()).start();
23      }
24
25      TCPSkravler_2(Socket s) {
26
27          forbindelse = s;
28      }
29
30      public void run() {
31
32          PrintWriter utskriver;
33          Scanner innleser;
34
35          try {
36              utskriver = new PrintWriter(forbindelse.getOutputStream());
37              innleser = new Scanner(forbindelse.getInputStream());
38
39              while(innleser.hasNextLine()) {
40                  utskriver.format("Du sa:\t\"%s\".\n", innleser.nextLine() )
                        ;
41                  utskriver.flush();
42              }
43
44          } catch (IOException e) {
45              e.printStackTrace();
46          }
47
48          finally {
49              try {
50                  forbindelse.close();
51
52              } catch (IOException e) {
53                  e.printStackTrace();
54              }
55
56          }
57      }
58  }
```

2.3 Håndtering av kappløp (race condition)

2.3.1 Eksempler

Listing 2.7: eksempler/02/Balanse_1.java

```
1  public class Balanse_1 extends Thread {
2
3  /*
4   * Eksempel: Traader med delt minne
```

```
5   * Del 1 av 3: Introduksjon av eksemplet
6   *
7   */
8
9       static final int T = 1000; // Totalsummen i spillet
10      static final int N = 5;    // Antall konti
11      static private int konto[] = new int[N];
12
13      java.util.Random slumptall = new java.util.Random();
14
15      public static void main(String[] args)  {
16
17          opprettKonti();
18          skriv_saldo();
19
20          new Balanse_1().start();
21
22          while(true) {
23
24              try {
25                  sleep(1000);
26              } catch (InterruptedException e) {
27                  e.printStackTrace();
28              }
29
30              skriv_saldo();
31          }
32      }
33
34      public void run()  {
35
36          int til, fra, belop;
37
38          while(true) {
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
49      private static void opprettKonti() {
50
51          int i;
52          for (i=0; i<N; i++)
53              konto[i]=T/N;
54      }
55
56      private static void skriv_saldo() {
57
58          int sum,i;
59
60          for (i=0, sum=0; i<N; i++) {
61
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 2.8: eksempler/02/Balanse_2.java

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

```
50      }
51
52      private static void opprettKonti() throws
53          InterruptedException {
54
55          int i;
56
57          for (i=0; i<N; i++)
58              konto[i]=T/N;
59      }
60
61      private static void skriv_saldo() throws
62          InterruptedException {
63
64          int sum, i;
65
66          for (i=0, sum=0; i<N; i++) {
67              System.out.printf("konto_%d:\t%d\n", i, konto[i]);
68              sum+=konto[i];
69          }
70
71          System.out.printf("--------------\nTotalt:\t%d\n==========\n",sum
                );
72      }
73  }
```

Listing 2.9: eksempler/02/Balanse_3.java

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

```
32          for (i=0; i<S; i++)
33              new Balanse_3().start();
34      }
35
36      public void run()  {
37
38          while(true)
39              flyttPenger();
40      }
41
42      private static synchronized void flyttPenger() {
43
44          int til, fra, belop;
45
46          fra=Math.abs( slumptall.nextInt() ) %N;
47          til=Math.abs( slumptall.nextInt() ) %N;
48          belop=Math.abs( slumptall.nextInt() ) % ( (konto[fra]/10) );
49
50          konto[fra]-=belop;
51          konto[til]+=belop;
52      }
53
54      private static void opprettKonti() throws
55          InterruptedException {
56
57          int i;
58
59          for (i=0; i<N; i++)
60              konto[i]=T/N;
61
62          skriv_saldo();
63      }
64
65      private static synchronized void skriv_saldo() throws
66          InterruptedException {
67
68          int sum, i;
69
70          for (i=0, sum=0; i<N; i++) {
71              System.out.printf("konto_%d:\t%d\n", i, konto[i]);
72              sum+=konto[i];
73          }
74
75          System.out.printf("--------------\nTotalt:\t%d\n==============\n",sum
                );
76          System.out.flush();
77      }
78  }
```

2.4 Øvelsesoppgaver

2.4.1 2.1

Forsøk å skrive programmet UDPSensor fra forrige forrige gang, slik at den får samme funksjonalitet som TCPSkravler_2.

2.4.2 2.2

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

2.4.3 2.3

- 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
3  /*
4   * Som TCPSkravler_2, men svarer med tilfeldig antall linjer.
5   * Tilleggne er markert.
6   */
7
8     public   java.net.Socket forbindelse;
9
10     public static void main(String[] args) throws java.io.IOException {
11
12         java.net.ServerSocket tjenerSocket = new java.net.ServerSocket
               (8888);
13         tjenerSocket.setReuseAddress(true);
14
15         while (true)
16             new TCPSkravler_3(tjenerSocket.accept()).start();
17     }
18
19     TCPSkravler_3(java.net.Socket s) {
20
21         forbindelse = s;
22     }
23
24     public void run() {
25
26         java.io.PrintWriter utskriver;
27         java.util.Scanner innleser;
28         int i;
29
30         try {
31             utskriver = new java.io.PrintWriter(forbindelse.getOutputStream
                   ());
32             innleser = new java.util.Scanner(forbindelse.getInputStream());
33
34             while(innleser.hasNextLine()) {
35
36     // Tillegg start
37                 for (i=0; i<(int)(Math.random()*3); i++)
38                     utskriver.println("Hei, hei");
39     // Tillegg slutt
40
41                 utskriver.format("Du sa:\t\"%s\".\n", innleser.nextLine() )
                       ;
```

```
42                  utskriver.flush();
43              }
44
45          } catch (java.io.IOException e) {
46              e.printStackTrace();
47          }
48
49          finally {
50              try {
51                  forbindelse.close();
52
53              } catch (java.io.IOException e) {
54                  e.printStackTrace();
55              }
56          }
57      }
58  }
```

Kapittel 3

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

3.1 Repetisjon

- Hvordan kan lager vi en tråd i java?
- Hvordan kan vi unngå kappløps-situasjoner i java?

3.2 Løsningsforslag på forrige øvelsesoppgave

3.2.1 2.2

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

Listing 3.1: 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 3.2: 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  }
```

3.2.2 2.3

- 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 flere linjer. Ved å kjøre lesing og skriving i hver sin tråd, blir de uavhengige av hverandre – de trenger ikke vente på tur.

Listing 3.3: 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         new Lytter(s).start();
14
15         while (inn.hasNextLine()) {
16             ut.println(inn.nextLine());
17             ut.flush();
18         }
19     }
20 }
```

Listing 3.4: 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 }
```

3.3 Mer trådsynkronisering

- Eksemplet TraadBryter.java demonstrerer 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  /*
2   * Haandtering av traadkjoering med
3   */
4
5  public class TraadBryter extends Thread {
6
7      private boolean kjor;
```

```
8       static final int antTr = 3;
9
10      public static void main(String[] args) {
11
12          TraadBryter[] traader;
13          java.util.Scanner inn;
14          int i, n;
15
16          traader = new TraadBryter[antTr];
17
18          // instansierer objekter og "starter" dem
19          for (i=0; i<antTr; i++) {
20              traader[i] = new TraadBryter();
21              traader[i].setName(String.format("Tr. %d", i));
22              traader[i].start();
23          }
24
25          inn = new java.util.Scanner(System.in);
26
27          while (inn.hasNext()) {
28
29              // leser traadnummer fra standard inngang
30              n = inn.nextInt();
31
32              // inverterer den boolske variablen kjor
33              // hos traad nummer n
34
35              if (traader[n].getKjor() == false)
36                  traader[n].setKjor(true);
37              else
38                  traader[n].setKjor(false);
39          }
40      }
41
42      public synchronized void run() {
43          try {
44              kjor = true;
45              while (true) {
46                  wait(1500); // suspenderer traaden i maksimum 1500 ms
47                  if (kjor) {
48
49                      System.out.printf("%s\n",this.getName());
50
51                  } else
52                      wait(); // suspenderer traaden
53              }
54
55          } catch (InterruptedException e) {
56              e.printStackTrace();
57          }
58      }
59
60      public synchronized void setKjor(boolean k) {
61          this.kjor = k;
62          if (kjor)
63              notify(); // aktiverer seg selv
64      }
65
```

```
66      public synchronized boolean getKjor() {
67          return this.kjor;
68      }
69  }
```

Ikke-blokkerende tjener

- Eksemplet NonBlockSkravler.java demonstrerer 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 til 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              it = hendelsesNokler.iterator();
52
53              while(it.hasNext()) {
54
55                  // Henter referanse til neste nøkkel
56                  nokkel = (SelectionKey) it.next();
57
58                  // Henter et heltall med flagg som indikerer hvilke
                        operasjoner
59                  // som er registert som klare i nøkkelen
60                  operasjoner = nokkel.readyOps();
61
62                  // Sjekker om flagget som indikerer oprasjonen 'accept er
                        satt
63                  if ((operasjoner & SelectionKey.OP_ACCEPT) == SelectionKey.
                        OP_ACCEPT) {
64                      tilkoble(nokkel);
65                      continue;
66
67                  }
68
69                  // Sjekker om flagget som indikerer oprasjonen 'read' er
                        satt
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,
118        * så vi må klargjøre dette før vi kaller på read().
119              */
120
121        java.nio.ByteBuffer buffer = java.nio.ByteBuffer.allocate(2048);
122        buffer.clear();
123
124        if (sC.read(buffer) == -1)
125            sC.socket().close();
126
127        buffer.flip(); // limit=posisjon; posisjon=0;
128
129        /*
130         * En ikke-blokkerende SocketChannel vil skrive til
131         * utgående socket-buffer er fullt
132         * og så returnere.
133         *
134         * write() må derfor av og til kalles flere ganger.
135         */
136
```

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

3.4 URL

- Uniform Resource Locator
- Definerer hvor på internett og hvordan
- <protokoll>://<vertsnavn>[:<port>]/absolutt_sti[?argumenter]
- Eksemplet WebLaster.java demonstrerer 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  /*
2   * Nedlasting av fil med HTTP
3   */
4
5  public class WebLaster {
6
7      public static void main(String[] args) throws
8
9          java.net.MalformedURLException,
10         java.io.IOException {
11
12         java.io.InputStream innstrom;
13         java.util.Scanner skanner;
14         java.net.URL url;
15
16         url       = new java.net.URL("http://www.hive.no");
17         innstrom  = url.openStream();
18         skanner   = new java.util.Scanner(innstrom);
19
20         while (skanner.hasNext())
21             System.out.println(skanner.nextLine());
22
23     }
24
25 }
```

3.5 Lesing av fil

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

Listing 3.8: eksempler/03/Brukere.java

```
/*
 * Lesing og "parsing" av csv-fil
 */

public class Brukere {

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

        java.util.Scanner innleser;
        String [] post;
        java.io.File filnavn;

        filnavn =  new java.io.File("/etc/passwd");
        innleser = new java.util.Scanner(filnavn);

        while (innleser.hasNextLine()) {
            post = innleser.nextLine().split(":");
            if (post.length > 4) {
                System.out.printf(
                    "Navn:\t%s\nLogin:\t%s\n\n",
                    post[4],
                    post[0] );
            }
        }
        innleser.close();
    }
}
```

3.6 Kommandolinje-argumenter

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

Listing 3.9: eksempler/03/Bruker.java

```
/*
 * Bruk av kommandolinjeargumenter
 */

public class Bruker {
    public static void main(String[] args) throws java.io.IOException {

        java.util.Scanner innleser;
        String [] post;
        java.io.File filnavn;

        if (args.length < 1) {
            System.out.println("Du maa oppgi et brukernavn som argument.");
            System.exit(1);
        }

        filnavn = new java.io.File("/etc/passwd");
        innleser = new java.util.Scanner(filnavn);

        while (innleser.hasNextLine()) {
            post = innleser.nextLine().split(":");
```

```
22              if ( post.length > 4 && args[0].equals(post[0]) ) {
23                  System.out.printf(
24                          "Navn:\t%s\nLogin:\t%s\n\n",
25                          post[4],
26                          post[0] );
27              }
28          }
29          innleser.close();
30      }
31  }
```

3.7 Øvelsesoppgaver

3.7.1 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.

3.7.2 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.

3.7.3 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 Repetisjon

- Hvordan kan vi kontrollere kjøringen av tråder i java?

4.1.1 Ikke-blokkerende IO

- Hvordan skiller det seg fra bruk av tråder?
- Hvorfor?
- Hvordan?

4.2 Løsningsforslag på forrige øvelsesoppgave

4.2.1 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  }
```

4.2.2 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("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  }
```

4.2.3 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 }
```

4.3 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      // Til brukes i PersonHenter_2 og PersonLagrer_2
4
5      String navn;
6      int id;
7
8      public Person(int d, String s) {
9          navn = s;
10         id   = d;
11     }
12
13     public String toString() {
14         return String.format("%d:\t%s\n", id, navn);
15     }
16 }
```

Listing 4.7: eksempler/04/PersonLagrer_1.java

```
1  // Demonstrasjon av skriving til "random access file"
2
3  public class PersonLagrer_1 {
4
5
6      public static void main(String[] args)
7      throws java.io.IOException {
8
9          final int fastBredde = 8;
10
11         java.io.RandomAccessFile fil = new java.io.RandomAccessFile("person
               .dat","rw");
12         java.util.Scanner inn= new java.util.Scanner(System.in);
13         int lengde,j;
14         String linje;
15
16         int i=0;
17
18         while(inn.hasNext()) {
19
20             fil.writeInt(i++);
21             linje = inn.nextLine();
22             lengde = linje.length();
23
24             if ( lengde <= fastBredde ) {
25                 fil.writeChars(linje);
26                 for (j=lengde; j<fastBredde; j++ )
27                     fil.writeChar('␣');
28             } else
29                 fil.writeChars(linje.substring(0, fastBredde));
30         }
31     }
32 }
```

Listing 4.8: eksempler/04/PersonHenter_1.java

```
1  // Demonstrasjon av lesing fra "random access file"
2
3  public class PersonHenter_1 {
```

```
4
5       public static void main(String[] args) throws java.io.IOException {
6
7           final int fastBredde = 8;
8           final int postStr = 4 + fastBredde*2;
9
10          StringBuffer buf;
11          long postNr;
12          int id, i;
13
14          java.io.RandomAccessFile fil = new java.io.RandomAccessFile("person
                .dat","r");
15          java.util.Scanner inn = new java.util.Scanner(System.in);
16          long antPost = fil.length()/postStr;
17
18          while (inn.hasNext()) {
19
20              buf  = new StringBuffer();
21              postNr = inn.nextInt();
22
23              if (postNr >= antPost)
24                  System.out.printf("Post␣%d␣finnes␣ikke.\n", postNr);
25
26              else {
27
28                  fil.seek(postNr*postStr);
29                  id = fil.readInt();
30
31                  for (i=0; i<fastBredde; i++)
32                      buf.append(fil.readChar());
33
34                  System.out.printf("%d:␣%s\n", id, buf);
35              }
36          }
37      }
38  }
```

4.4 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   // Demonstrerer skriving av serialiserte objekter til fil
2
3   public class PersonLagrer_2 {
4
5       public static void main(String[] args) throws java.io.IOException {
6
7           java.io.ObjectOutputStream ut = new java.io.ObjectOutputStream(
8               new java.io.FileOutputStream("person_2.dat"));
9
10          java.util.Scanner inn = new java.util.Scanner(System.in);
11
12          int i=0;
13
```

```
14          while(inn.hasNext())
15              ut.writeObject(new Person(i++, inn.next()));
16
17          ut.close();
18      }
19  }
```

Listing 4.10: eksempler/04/PersonHenter_2.java

```
1  // Demonstrerer lesing av serialiserte objekter fra fil
2
3  public class PersonHenter_2 {
4
5      public static void main(String[] args)
6      throws java.io.IOException, ClassNotFoundException {
7
8          java.io.ObjectInputStream inn = new java.io.ObjectInputStream(
9              new java.io.FileInputStream("person_2.dat"));
10
11         while (true) {
12             try {
13                 System.out.println( inn.readObject() );
14             }
15             catch (java.io.EOFException e) {
16                 inn.close();
17                 break;
18             }
19         }
20     }
21 }
```

4.5 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      @SuppressWarnings ("unchecked")
4
5      public static void main(String[] args)
6      throws java.io.IOException, ClassNotFoundException {
7
8          java.io.ObjectInputStream inn;
9          java.util.Vector <Person> pv;
10
11         inn = new java.io.ObjectInputStream(
12             new java.io.FileInputStream("person_3.dat"));
13
14         pv = (java.util.Vector<Person>) inn.readObject();
15
16         for (Person p:pv)
17             System.out.println(p);
18
19         inn.close();
20     }
21 }
```

4.6 Øvelsesoppgaver

4.6.1 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).

4.6.2 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.

Del II

Web

Kapittel 5

HTML,CGI

5.1 Repetisjon

5.1.1 Random Access

- Hva menes med "random access" i forbindelse med filtilgang?
- Gi et eksempel på at det kan være nyttig å bruke slik fil-tilgang.
- Gi et eksempel hvor det ikke er mulig å utnytte muligheten som slik filtilgang gir.
- Hva heter metoden for å flytte skrive/lese-markøren i en fil som er åpnet for med slik tilgang?

5.1.2 Serialisering

- Hva menes med serialisering av objekter?
- Hvordan oppnår vi det i java?
- Hvorfor skal vi serialisere?

5.2 Løsningsforslag til oppgaver fra forrige gang

5.2.1 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-storrelse
8         final int postStr = 4 + fastBredde*2;
9
```

```
10          if (args.length<2)
11              System.exit(1); // Avslutt med feilkode
12
13          // Tolk forste argument som et heltall
14          int             id = Integer.parseInt(args[0]);
15
16          // AApne 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          // Gjennomloper 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  }
```

5.2.2 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          // Blir kvitt linjeskiftet
20              inn.nextLine();
21
22              // Leser objekt og skriver det ut.
23              System.out.println( sInn.readObject() );
24          }
25      }
26  }
```

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 traaden. 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) { // Soker basert paa 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  }
```

5.3 Grunnleggende HTML

5.3.1 Historisk perskpektiv

`http://www.w3.org/History/19921103-hypertext/hypertext/WWW/MarkUp/Tags.html`

- ..

`http://www.w3.org/TR/html5/`

5.3.2 HTML

- HyperText Markup Language
- "tagge-språk" for web-dokumenter
- Basert på SGML
- Utviklet på CERN (Berners-Lee)
- Standarisert av IETF (Internet Engineering Task Force) i 1995

`http://www.ietf.org/rfc/rfc1866`

Utviklet med tanke på plattformuavhengighet

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

Eksempel

- 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>
    <meta http-equiv="content-type" content="text/html; charset=UTF-8">
    <title> Tittel </title>

  </head>


  <body>

    <h1> Overskrift </h1>

    Vanlig tekst.
    Linjeskift m&aring; 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>

    <p>
      <a href="http://validator.w3.org/check?uri=referer">
  <img
      src="http://www.w3.org/Icons/valid-html401"
      alt="Valid HTML 4.01 Transitional"
      height="31"
      width="88"
      >
      </a>
  </p>


  </body>

</html>
```

5.3.3 HTML 5

- HTML 5 - i ferd med å implementeres
- HTML 5 skal erstatte HTML 4.01
- Er ikke lenger basert på SGML
- Forsøk på å avskaffe behovet for tilleggs-moduler (”plugins”).
- Standarden inkluderer DOM m/API

Inneholder blant annet egne tagger for lyd og video

- video
- audio
- canvas

Semantiske tagger

- section
- article
- header
- nav

støtte for

- SVG (vektorgrafikk)
- MathML

Eksempler

Listing 5.5: eksempler/05/grunnleggende2.html

```
<!doctype html>

<html>

  <head>
    <meta charset="utf-8">
    <title> Tittel </title>

  </head>


  <body>

    <h1> Overskrift </h1>

    Vanlig tekst.
    Linjeskift m&aring; 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>

    <p>
      <a href="http://validator.w3.org/check?uri=referer">
        This document was successfully checked as HTML5!
```

```
30          </a>
31      </p>
32
33
34      </body>
35
36  </html>
```

Listing 5.6: eksempler/05/lyd.html

```
1  <!doctype html>
2
3  <html>
4
5      <head>
6          <meta charset="utf-8">
7          <title>Lydtest</title>
8
9      </head>
10
11     <body>
12
13         <h1>Lydtest</h1>
14
15         <audio controls="controls">
16
17             <source
18         type="audio/ogg"
19         src="01.ogg"
20         />
21
22         </audio>
23     </body>
24 </html>
```

5.4 HTTP

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

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

5.4.1 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
- Fra versjon 1.1 kan punkt 2 og 3 gjentas flere ganger før TCP-forbindelsen stenges.

5.4.2 tilstandsløs

- Tjeneren lagrer ingen informasjon om klientenes forespørsler
- Vanskelig å støtte konseptet 'sesjon' som er nødvendig for å støtte (atomiske) transaksjoner.

5.4.3 HTTP-forespørsel

Hode (header)

- forespørsel-type (f.eks GET, POST, HEAD)
- navn/sti på "ressurs" (f.eks /losninger/03/logg)
- HTTP-versjonen (f.eks. HTTP/1.1)

eventuelt kropp

adskilt fra hodet med en tom linje

5.4.4 HTTP-respons

Hode

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

eventuelt kropp

adskilt fra hodet med en tom linje

5.4.5 Demonstrasjon av http-transaksjon

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

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

5.5 Grunnleggende CGI

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

5.5.1 Web-tjener vedlikeholder en del miljøvariabler som skriptet arver

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

Listing 5.7: eksempler/05/env.cgi

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

Listing 5.8: eksempler/05/Env.java

```
public class Env {

    public static void main(String[] args) {

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

        for (java.util.Map.Entry<String, String> e : System.getenv().
            entrySet() )
            System.out.println(e);
    }
}
```

5.5.2 Hallo_1

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

Listing 5.9: eksempler/05/hallo_1.html

```
<!doctype html>
<html>
  <head>
      <meta charset="utf-8">
      <title>Hallo_1</title>
  </head>

  <body>

    Si hallo til:
    <ul>

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

    </ul>

  </body>
</html>
```

Listing 5.10: eksempler/05/hallo_1.cgi

```
#!/bin/sh

java -Dfile.encoding=UTF-8 Hallo_1
```

Listing 5.11: eksempler/05/Hallo_1.java

```
public class Hallo_1 {

    public static void main(String[] args) {

        // Skriver ut 'http-header' for 'plain-text'
        System.out.printf("Content-type:text/plain;charset=utf-8\n\n");

        // Skriver ut 'http-body'
        System.out.printf(
            "Hallo␣%s.\nHa␣en␣fin␣dag␣:)",
            System.getenv("QUERY_STRING")
        );

    }
}
```

5.5.3 Hallo_2

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

Listing 5.12: eksempler/05/hallo_2.html

```
<!doctype html>
<html>
  <head>
      <meta charset="utf-8">
      <title>Hallo_2</title>
  </head>
  <body>
    <form action=hallo_2.cgi>
     <input type=text name=navn>
     <input type=submit>
    </form>
  </body>
</html>
```

Listing 5.13: eksempler/05/hallo_2.cgi

```
#!/bin/sh

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

Listing 5.14: 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");
```

```
6
7           System.out.printf(
8                "Variabelen QUERY_STRING:\t%s",
9                System.getenv("QUERY_STRING")
10          );
11
12      }
13  }
```

5.5.4 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 5.15: eksempler/05/hallo_3.html

```
1  <!doctype html>
2  <html>
3    <head>
4         <meta charset="utf-8">
5         <title>Hallo_3</title>
6    </head>
7    <body>
8      <form action=hallo_3.cgi>
9       <input type=text name=navn>
10      <input type=submit>
11     </form>
12   </body>
13 </html>
```

Listing 5.16: eksempler/05/hallo_3.cgi

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

Listing 5.17: 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>"+
12              "<html><head>"+
13       "<meta charset='utf-8'>"+
14       "<title>Hallo_3</title></head>"
15         );
16
17         // Skriver ut html-kroppen
18         System.out.printf(
```

```
19              "<body>Variabelen QUERY_STRING:\t%s</body>",
20              java.net.URLDecoder.decode(
21                  System.getenv("QUERY_STRING"),
22                  "UTF-8"
23              )
24          );
25
26          // Skriver ut avsluting av html-dokument
27          System.out.println("</html>");
28      }
29  }
```

5.5.5 Hallo_4

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

Listing 5.18: eksempler/05/hallo_4.cgi

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

Listing 5.19: 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>"+
16             "<html><head><meta charset='utf-8'><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                 "UTF-8"
32             ) +
```

```
33
34                  "</body></html>"
35              );
36
37          }
38      }
```

5.6 Litteratur

- Følgende seksjoner fra boka "Database Systems (Connoly & Begg,2010) er relevant å lese i forbindelse med denne forelesningen:
- 30.2.1
- 30.2.2
- 30.4

5.7 Øvelser

5.7.1 5.1

- Bli kjent med systemet.

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 debbie.
- 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.

5.7.2 5.2

Repetsisjonsoppgave: 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.

5.7.3 5.3

I eksemplet Hallo_4, forsvinner teksten fra skjemaets tekstfelt, når brukeren trykker "Send inn". Endre programmet slik at verdiene forblir i tekstfeltene.

5.7.4 5.4

a)

Lag en versjon av NavneEndrer (Oppgave 4.1) med et web-basert grensesnitt ved hjelp av CGI. Sørg for at Web-grensesnittet er et gyldig HTML5 dokument.

b)

Lag en versjon av PersonKlient (Oppgave 4.2) med et web-basert grensesnitt ved hjelp av CGI. Sørg for at Web-grensesnittet er et gyldig HTML5 dokument.

Kapittel 6

JavaScript

6.1 Løsningsforslag til oppgave fra forrige gang

6.1.1 5.2

- 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 {
32                  pv.addElement((Person) fInn.readObject());
33              }
34
35              catch (EOFException e) {
36                  fInn.close();
37                  break;
38              }
39          }
40
41          pvStr = pv.size();
42
43          while (true) {
44
45              s = ss.accept();
46
47              sUt = new ObjectOutputStream(s.getOutputStream());
48              sInn = new Scanner(s.getInputStream());
49
50              while(sInn.hasNext()) {
51
52                  id = sInn.nextInt();
53                  sInn.nextLine();
54                  pos = -1;
55
56                  for (i=0; i < pvStr; i++)
57
58                      if (pv.elementAt(i).id == id) {
59                          pos = i;
60                          break;
61                      }
62
63                  if (pos == -1)
64                      sUt.writeObject(new Person(-1,"-"));
65                  else
66                      sUt.writeObject(pv.elementAt(pos));
67
68              }
69
70              s.close();
71          }
72      }
73  }
```

6.2 Repetisjon

- HTTP
- HTML
- CGI

6.3 Intro

6.3.1 JavaScript/JScript/ECMAScript

- JavaScript er ikke java.

```
"ECMAScript was always an unwanted trade name
that sounds like a skin disease."
(e-post fra JavaScripts opphavsmann).
```

`http://bellard.org/jslinux/`

- Både klientside og tjenerside
- Mye brukt i Web2.0/AJAX/HTML5
- Objektorientert (prototypebasert/klasseløst)
- Dynamiske variabeltyper
- Java-lignende syntax

6.3.2 I nettleser

- Kan lese og skrive egenskaper i applets og plugins
- På android: Kan få tilgang til java-objekter.

6.3.3 Eksempel

Listing 6.2: eksempler/06/javascripthallo.html

```
1  <!doctype html>
2  <html>
3
4    <head>
5      <title>JavaScript-hallo</title>
6      <meta charset='UTF-8'>
7    </head>
8
9    <body>
10
11     <script type="text/javascript">
12       document.write("<h1 id='hilsen'>Hallo</h1>");
13       var overskrift = document.getElementById('hilsen');
14     </script>
15
16     <button id='knapp' onclick="overskrift.innerHTML='Borte'">
17       Klikk her
18     </button>
19
20   </body>
21
22 </html>
```

- konsoll
- rhino

6.4 Leksikal struktur

6.4.1 Store og små bokstaver

- Javascript skiller mellom store og små bokstaver.
- I nettlesere er hendelseshåndterere som er definert som et html-attributt
- Som et html-attributt skilles 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

6.4.2 "Statements"

Kommandosetninger (en: Statements) avsluttes med semikolon eller linjeskift.

6.4.3 Kodeblokk

- En enkelt kommandosetning
- Eller flere kommandosetninger omsluttet av krøllparenteser.

6.4.4 Kommentarer

Kommentarer skrives i koden som i C, C++ og Java.

6.4.5 "Free form"

Ekstra "whitespace" (mellomrom og tabulator og ekstra linjeskift) ingoreres.

6.4.6 Konstanter/"literals"

tallkonstanter

- 1.2
- 1.432e-5
- 8.9E+13

tekstkonstanter

- ”A”
- ’a’
- ’AB C’
- ”ab c”

objektkonstanter

- { navn:’per’, nummer:123 }
- kan brukes ved initialisering av objekter.

tabell-konstant

1,2,3,67

- Kan brukes ved initialisering av tabeller.

noen spesielle konstanter

- true
- false
- null

6.5 Variabler og datatyper

6.5.1 Variablenes typer settes dynamisk

- 1+2 -> 3
- 1+”2” -> ”12”

6.5.2 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

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

6.5.3 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.

6.5.4 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')

6.5.5 Boolske verdier

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

6.5.6 Rekker

var rekke1=[1,,,,9]

6.5.7 'null'

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

6.5.8 'Undefined'

'Undefined' er verdien til:

- variabel som ikke eksisterer
- variabel som ikke er initialisert

Følgende er sant:

undefined==null

6.6 flytkontroll

6.6.1 if ... else

- if (test) kodeblokk
- if (test) kodeblokk else kodeblokk

6.6.2 while

- while () kodeblokk
- do {} while ()

6.6.3 for

- for (init ; test ; endring) kodeblokk
- for (x in X) kodeblokk

6.6.4 switch

switch(x){case a: kodeblokk case b: kodeblokk default: kodeblokk}

6.6.5 continue

kan brukes i løkker

6.6.6 break

kan brukes i løkker og switch-case

6.7 Funksjoner

6.7.1 Eksempel på funksjon

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

6.8 JavaScript i Nettlesere

6.8.1 Objekthierariket

`http://en.wikipedia.org/wiki/Document_Object_Model`

- 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'.

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)

window.navigator.

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

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)

window.document.forms[][].

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

6.8.2 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>

HTML-kommentarer

- For hindre nettlesere som ikke kjenner script-taggen i å vise frem selve koden til brukerene, kan koden beskyttes av html-kommentarer.
- <!– tolkes som //
- –> blir ikke gjenkjent. Derfor skrives //–>

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>

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>
- <SCRIPT SRC="http://tulla.hivert.no/mineskript/diverse.js"><SCRIPT>

ARCHIVE

- Man kan pakke sammen flere JavaScript filer (og andre filer; f.eks. signaturer) i JAR(Java-arkiv)-filer.

Eksempel: <SCRIPT ARCHIVE="diverse.jar" SRC="kalkulasjoner.js"> </SCRIPT>

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();" >

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.

Eksempler: <A HREF="javascript: alert(Date())">nå <FORM ACTION="javascript: '<H1><CENTER>Hei og hopp</CENTER></H1>"'>

Eksempel

Listing 6.3: eksempler/06/document.write.html

```
<!DOCTYPE html>
<html lang='no'>
  <head>

    <title> Test av document.write()</title>
    <meta charset='utf-8'>

    <script>

    function skriv(){

  for (i=0; i<3; i++){

      document.write("<h1>Overskrift " + i + "</h1>");

      for (j=0; j<3; j++)
    document.write('<p>Avsnitt ', i, '.', j, '</p>');
  }
    }

    </script>
  </head>

  <body>
    <script>
    document.write(
  "Adressen til dette dokumentet er: <i>",
  window.location,
  "</i><br>"
    );
    </script>

    <button onclick='skriv()'>
      Test av document.write()
    </button>

    <button onclick='location="http://validator.w3.org/check?uri="+location
        '>
      Valider denne siden
    </button>

  </body>
</html>
```

6.8.3 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.

6.8.4 Ajax

- Asynchronous Java- Script and XML

XMLHttpRequest

- open()
- setRequestHeader()
- send()
- onreadystatechange

Eksempel

Listing 6.4: eksempler/06/ajax-sql-klient.html

```
<!doctype html>
<head>
<meta charset="utf-8">
<script>

function sporre(event){
    if (event.keyCode==13) {
  var f=new XMLHttpRequest();
  var s=document.getElementById("s").value;
  f.open("GET","sql-klient3.cgi?s="+s, true , "oop", "va");
  f.onload=function(){
      document.getElementById("tabell").innerHTML=f.responseText;
  };
  f.send(null);
    }
}
</script>
</head>

<body>

    <input
        size=80
        id=s
        type=text
        onKeyUp=sporre(event)
        >
    <div id="tabell"></div>

</body>
```

Listing 6.5: eksempler/06/sql-klient3.cgi

```
#!/bin/sh

export CLASSPATH=/usr/share/java/mysql-connector-java.jar:
java -D -Dfile.encoding=UTF-8 SqlKlient3
```

Listing 6.6: eksempler/06/SqlKlient3.java

```
public class SqlKlient3 {

    private static java.sql.ResultSetMetaData met;
    private static java.sql.Connection        lnk;
    private static java.sql.Statement         stm;
    private static java.sql.ResultSet         res;

    public static void main(String[] args)
    throws
        java.sql.SQLException,
        ClassNotFoundException {

        String url, drv, usr, pwd, sql;
        int kol, i;

```

```
16          url = "jdbc:mysql://localhost/bokbase";
17          drv = "com.mysql.jdbc.Driver";
18          usr = "db"; pwd = "ada";
19
20          Class.forName(drv);
21          lnk = java.sql.DriverManager.getConnection(url,usr,pwd);
22          stm = lnk.createStatement();
23
24          try {
25              sql = java.net.URLDecoder
26                    .decode(System.getenv("QUERY_STRING"), "UTF-8")
27                    .substring(2);
28          } catch (Exception e) {
29              sql = "";
30          }
31
32          res = stm.executeQuery(sql);
33          met = res.getMetaData();
34          kol = met.getColumnCount();
35
36          System.out.println("Content-type:text/html;charset=utf-8\n\n");
37          System.out.println("<table border=1><tr>");
38
39      // Utskrift av overskrifter
40          for (i=1; i<=kol; i++)
41              System.out.printf("<th>%s</th>", met.getColumnName(i));
42          System.out.println("</tr>");
43
44      // Utskrift av data
45          while(res.next()) {
46              System.out.println("<tr>");
47              for ( i = 1; i <= kol; i++ )
48                  System.out.printf("<td>%s</td>",res.getString(i));
49              System.out.println("</tr>");
50          }
51
52          System.out.println("</table>");
53      }
54  }
```

6.9 Oppgaver

6.9.1 6.1

a)

Lag et javascript som kjører lokalt og skriver "Hallo verden!" i nettleseren.

b)

Lag et javascript som kjører på debbie og skriver ut "Hallo verden" i nettleseren ved hjelp av CGI og rhino.

6.9.2 6.2

a)

- Gå gjennom JavaScript Tutorial "JS Basic" frem til "JS Try .. Catch"

`http://www.w3schools.com/js/default.asp`

b)

- Gå gjennom kapitlene fra og med 'XML Javascript' til 'XML Applications'

`http://www.w3schools.com/xml/xml_http.asp`

6.9.3 6.3

Lag en webside som lar brukeren skrive inn enkle funksjoner i et tekstfelt. F.eks. "x*x/400+50" Når brukeren trykker på en knapp, skal grafen plottes i på et lerret (canvas). Du kan bruke eval() for å beregne y verdiene ut fra innholdet i tekstfeltet.

Del III

Android

Kapittel 7

Android

7.1 Løsningsforslag

7.1.1 6.3

Oppgavetekst

Lag en webside som lar brukeren skrive inn enkle funksjoner i et tekstfelt. F.eks. "x*x/400+50" Når brukeren trykker på en knapp, skal grafen plottes i på et lerret (canvas). Du kan bruke eval() for å beregne y verdiene ut fra innholdet i tekstfeltet.

Løsningsforslag

Listing 7.1: losninger/06/graftegner.html

```
<!DOCTYPE html>
<html lang='no'>
  <head>
    <meta charset='utf-8'>
    <title>Lerretstegning </title>

    <script>

    function tegne() {

  lerret = document.getElementById('lerret')
  c = lerret.getContext('2d')
  c.beginPath()

  for (x=0.0; x<500; x+=10.0){

      y = lerret.height - eval(
    document.getElementById('funksjon').value
      )

      if (x == 0)
    c.moveTo(x,y)
      else
    c.lineTo(x,y)
  }

  c.stroke()
```

```
28      }
29      </script>
30    </head>
31
32    <body>
33
34      <button onclick='location="http://validator.w3.org/check?uri="+location
            '>
35        Valider
36      </button>
37
38      <canvas id="lerret" width="500" height="500">
39
40      </canvas>
41
42
43      <form action='javascript:tegne()'>
44
45        <label for='funksjon'> Funksjon </label>
46        <input type='text' id='funksjon'>
47
48        <input type='submit' id='tegne'>
49    </form>
50    </body>
51  </html>
```

7.2 Repetisjon

- Hva er Javascript?
- Hva brukes javascript til?
- Hva er DOM?
- Hva er XMLhttpRequest?

7.3 Android-pensum

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

7.3.1 Til denne forelesningen

Application Fundamentals

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

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

`http://developer.android.com/tools/projects/projects-eclipse.html`

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

`http://developer.android.com/tools/projects/projects-cmdline.html`

7.4 Hva er Android?

- Komplett programvare-stack
- OS, mellomvare og nøkkelapplikasjoner
- Basert på åpen kildekode

7.4.1 Eid av

- open handset alliance - konsortium av mange selskaper. 84 selskaper (pr. 12/3-2013).

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

7.4.2 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 Play, 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" - BSD-derivat av libc (Standard C system library) tilpasset integrerte ("embedded") linuxbaserte enheter.
- SSL

SQLite `http://sqlite.org/`

- relasjonsdatabase-motor

WebKit `http://webkit.org/`

- nettlesermotor (web browser engine)
- brukes av mange populære nettlesere som Safari, mobile Safari, Google Chrome og selvfølgelig androids egen nettleser.

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).

7.5 Installere utviklingsmiljø

7.5.1 SDK Tools

Pass på å få med 'tools' og 'platform-tools' med i søksetien.

7.5.2 Plattformer

Ta med alle versjoner du ønsker å teste dine applikasjoner i.

7.5.3 Eclipse-plugin

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

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

7.6 Opprette AVD (Android Virtual Device)

7.6.1 Fra GUI

start 'android' enten fra kommondolinjen eller fra menyvalg i eclipse.

7.6.2 Fra kommandolinjen

- 'android create avd'

Påkrevde parametre

-n navnet du vil gi den virtuelle enheten

-t

- "target ID"
- hvilken verjson av android som skal installeres på den virtuelle enheten.

Andre parametre

Kommandoen 'android –help create avd' gir en oversikt over alle parametrene.

eksempel

- Eksemplet under lager en virtuell enhet med Android 2.2 med navnet testAndroid-VirtualDevice og et med et virutelt minnekort på 64MB. Argumentet -f, sørger for at testAndroidVirtualDevice blir overskrevet dersom den allerede finnes.

```
android create avd -t 4 -n testAndroidVirtualDevice -f -c 64M
```

7.6.3 Starte emulator

- emulator -avd minAVD
- eller (fra android-GUI)
- Eksempel: emulator -shell -show-kernel @minAVD

7.7 Android Applikasjon

- Skrevet i Java
- Kompileres til en androidpakke
- Androidpakkene har fil-etternavn ".apk"

7.7.1 Applikasjonene blir "sandkasset" (sandboxed) på enheten

- applikasjonene får hver egen linux-bruker
- applikajsonene kjører i hver sin virtuelle maskin

```
# ps
USER     PID   PPID  VSIZE RSS     WCHAN    PC         NAME
root     1     0     296   204     c008de04 0000c74c S /init
root     2     0     0     0       c004b334 00000000 S kthreadd

...

root     569   1     3332  148     ffffffff 0000e8c4 S /sbin/adbd
system   583   560   201048 25564  ffffffff afe0c45c S system_server
radio    623   560   109928 17560  ffffffff afe0d3e4 S com.android.phone
app_3    627   560   108048 18352  ffffffff afe0d3e4 S android.process.acore
app_15   648   560   105772 13196  ffffffff afe0d3e4 S com.android.mms
app_0    666   560   94292 12604   ffffffff afe0d3e4 S com.android.alarmclock
app_4    675   560   95408 13536   ffffffff afe0d3e4 S android.process.media
app_3    698   560   96328 12920   ffffffff afe0d3e4 S com.android.inputmethod.latin
root     739   554   888   340     00000000 afe0c1bc R ps
```

- "principle of least privilege"
- to applikasjoner kan i visse tilfeller dele bruker-id, og dermed få tilgang til delte resursser
- Applikasjon kan be brukeren om tilgang til IO-data. Dette må skje ved installasjon av applikasjonen.

7.7.2 Applikasjons-komponenter

- En android-applikasjon er bygd opp av komponenter.
- Hver komponent må deklareres i en manifest-fil
- Hver komponent kan være inngangspunkt (entry point) – ingen main()
- Det er fire ulike komponent-typer med hver sin hensikt og hver sin type livs-syklus.

Fire ulike komponentyper

Activity

- Representerer et skjermbilde med et brukergrensesnitt.
- Er en uavhengig komponent som kan kalles av andre applikasjoner (dersom eier-applikasjonen tillater det). På den måten kan den utgjøre et ingangspunkt.
- Implementeres som en subklasse av android.app.Activity
- Aktiveres med en asynkron melding kalt 'intent'

Starte en aktivitet:

- objekt av klassen 'Intent' som arguement til 'startActivity()'
- objekt av klassen 'Intent' som arguement til 'startActivityForResult()'
- To typer Intent:Eksplisitt/implisitt Intent (app/tjeneste)

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

Services

- Kjører i bakgrunnen
- Har ikke bruker-grensesnitt
- Implementeres som en subklasse av android.app.Service
- Aktiveres med en asynkron melding kalt 'intent'

Starte en service:

- objekt av klassen 'Intent' som arguement til 'startService()'
- objekt av klassen 'Intent' som arguement til 'bindService()'

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

Broadcast receivers

- Reagerer på system-kringkastede meldinger (broadcasts).
- Kringkastede medlinger kan komme fra systemet (f.eks. lavt batterinivå) eller fra applikasjoner.
- Har ikke eget brukergrensesnitt, men kan varsle i statusfeltet.
- Implementeres som subklasse av android.content.BroadcastReceiver
- Aktiveres med en asynkron melding kalt 'intent'

Sende en broadcast:

- objekt av klassen 'Intent' som arguement til 'sendBroadcast()'
- objekt av klassen 'Intent' som arguement til 'sendOrderedBroadcast()'
- objekt av klassen 'Intent' som arguement til 'sendStickyBroadcast()'

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

Content providers

- Håndterer data som skal brukes av flere applikasjoner
- Eks: En 'content provider' håndter kontaktene som er lagret på android-enheten.
- metoden 'query()' utføres på et objekt av klassen 'ContentResolver'.

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

7.7.3 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

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

Eksempel:

```
android create project -t 4 -p testkat -k tn.testpakke -a testaktivitet
cd testkat
ant debug
cp bin/*-debug.apk ~/public_html/apks/
```

installere fra PC på usb-tilkoblet android-enhet

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

installere fra PC på virtuell android-enhet

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

Notat 1

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

Listing 7.2: 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 }
```

7.8 Øvelser

7.8.1 7.1

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

7.8.2 7.2

- a) Installer en av app'ene du lagde på en android-enhet ved hjelp av verktøyet adb
- b) Installer en av app'ene du lagde ved å laste dem ned til en android-enhet fra en web-server (f.eks. fra oopva60.hive.no/ <DIN BRUKERID>/apks/

Kapittel 8

Android og javascript

8.1 Repetisjon

- Hva er android?
- Hvilke komponent-typer finner vi i en android applikasjon?
- Forklar hvordan androidapplikasjonene blir "sandkasset".
- Hvordan tildeles rettigheter til en app?

8.2 Androidpensum til denne forelesningen

8.2.1 Web Apps Overview

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

8.2.2 Building Web Apps in WebView

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

8.3 HTML for android plattformen

- Android kategoriserer skjermer basert på punkt-tetthet.
- Tetthet(density) måles i antall punkter pr. tomme (dpi - dot per inch).

8.3.1 Ulike kategorier av tetthet støttes i plattformen

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

8.3.2 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]
">
```

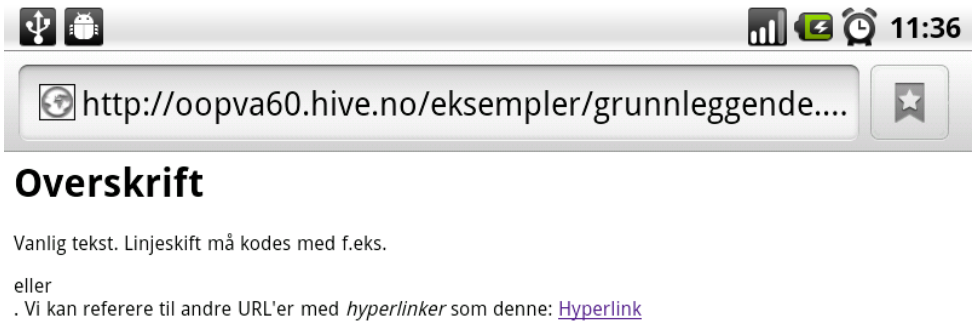
8.3.3 Eksempel

Listing 8.1: eksempler/05/grunnleggende.html

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

<html>

  <head>
    <meta http-equiv="content-type" content="text/html; charset=UTF-8">
    <title> Tittel </title>

  </head>


  <body>

    <h1> Overskrift </h1>

    Vanlig tekst.
    Linjeskift m&aring; 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>

    <p>
      <a href="http://validator.w3.org/check?uri=referer">
  <img
      src="http://www.w3.org/Icons/valid-html401"
      alt="Valid HTML 4.01 Transitional"
      height="31"
```

```
33          width="88"
34          >
35            </a>
36      </p>
37
38
39      </body>
40
41  </html>
```

eksempler/05/grunnleggende.html



Listing 8.2: eksempler/05/mobil.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> Tittel </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> Overskrift </h1>
22     <p>
23       Vanlig tekst.
24       Linjeskift maa kodes med f.eks. &lt;p&gt; eller &lt;br&gt; <br/>.
25       Vi kan referere til andre URL'er med <i>hyperlinker</i> som denne:
```

```
26          <a href="http://www.hive.no">
27      Hyperlink
28          </a>
29        </p>
30      </body>
31
32  </html>
```

eksempler/mobil.html

8.4 Web-applikasjon

8.4.1 To måter å levere applikasjoner til android-plattformen

- 1."Native"-applikasjon
- 2. Web-applikasjon

`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.

8.4.2 Eksempler

Hallo web

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

Listing 8.3: 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 8.4: 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 }
```

8.5 Litt mer javascript

8.5.1 dialogbokser

alert()

- alert(tekststreng)
- med OK-knapp

confirm()

- confirm(tekststreng)
- to knapper OK og Cancel
- returnerer true eller false

prompt()

- prompt(tekststreng, standardverdi)
- to knapper OK og Cancel
- et tekstfelt ferdig utfylt med standardverdi
- returnerer 'null' ved avbryt, ellers returneres verdien av tekstfeltet.

8.5.2 feilhåndtering

try - catch

try blokk catch(err) blokk

throw

- throw tekststreng
- catch(err) if err==tekststreng ...

8.5.3 objekter

- egenskaper
- metoder

opprette objekter

new Object()

konstruktør

- function objektnavn(arg1, arg2, ..) { this.egenskap=verdi; ... }
- new objektnavn(arg1, arg2, ..)

8.6 javascript koblet med java objekter (på android)

8.6.1 fra java-objekt til html-side

Listing 8.5: eksempler/08/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 8.6: eksempler/08/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 8.7: eksempler/08/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>
```

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

Listing 8.8: eksempler/08/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 8.9: eksempler/08/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  // WebChromeClient kalles ved hendelse som paavirker visning av
        brukergrensesnitt,
26  // som f.eks. meldingsbokser eller fremdrifts indikator.
27  class MinWebChromeKlient extends android.webkit.WebChromeClient {
28
29    @Override
30    public boolean onJsAlert(
31        android.webkit.WebView view,
32        String url,
33        String message,
34        android.webkit.JsResult result) {
35
36      this.visToast(message);
37      result.confirm();
38      return true;
39    }
40
41    public void visToast(String tekst) {
42
43      android.widget.Toast.makeText(
44          kontekst,
45          tekst,
46          android.widget.Toast.LENGTH_SHORT
47      ).show();
48    }
49
50    public void skrivUt(final String tekst) {
51
52      // Legger "runnable" i medlingskoe
53      mHandler.post(
54          // Kjoerer kode i en ny traad
55          new java.lang.Runnable() {
56            public void run() {
57
58              wv.loadUrl("javascript:ut.innerHTML='"+tekst+"'");
59            }
60          });
61    }
```

```
62   }
63 }
```

8.7 Øvelser

8.7.1 8.1

a)

- 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`

b)

Lag et cgi-program som prøver å gjette om klienten kjører på en mobil plattform. Dersom den gjetter pc skal den levere filen 'eksempler/grunnleggende.html' ellers skal den levere 'eksempler/mobil.html'

8.7.2 8.2

Gjennomgå følgende javascript-tutorials:

a) `http://www.w3schools.com/js/js_popup.asp`

b) `http://www.w3schools.com/js/js_errors.asp`

c) `http://www.w3schools.com/js/js_objects.asp`

8.7.3 8.3

- 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 lokalt på android-enheten. Utskriften skal komme til syne i html-siden sammen med tekstfeltet og knappen. Under finner du et kodeeksempel som demostrerer hvordan en prosess startes i Java.

Oppstart av prosess

Listing 8.10: eksempler/08/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 }
```

8.7.4 8.4

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`

8.7.5 8.5

- App'en i 8.4 og Notat02 går tilbake til forrige aktivitet når du trykker på tilbakeknappen. 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`

Del IV

Java på tjenersiden

Kapittel 9

Servlets

9.1 Repetisjon

9.1.1 CSS

- Hva brukes CSS til?
- Beskriv syntaksen til CSS.
- Hvor plasseres CSS koden?
- Hva brukes klasser til i CSS?

9.1.2 Androidkomponenten Service

- Hva brukes Service til?
- Hva menes med 'foreground' i forbindelse med Service?
- Hvordan kommuniserer en Service med brukeren?

9.2 Løsningsforslag på 09.1

9.2.1 Oppgavetekst

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

9.2.2 Løsningsforslag

- Følgende filer inneholder endringer:

Listing 9.1: losninger/09/09.1/ServiceTestMedPause/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
12          if (mp==null)
13            mp = android.media.MediaPlayer.create(this, R.raw.forelesning);
14
15          if(mp.isPlaying())
16            mp.pause();
17
18          else
19            mp.start();
20      }
21
22      @Override
23      public void onDestroy() {
24          super.onDestroy();
25          mp.release();
26      }
27
28      @Override
29      public android.os.IBinder onBind(android.content.Intent intent) {
30          return null;
31      }
32
33  }
```

Listing 9.2: losninger/09/09.1/ServiceTestMedPause/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     <script>
15
16     function pause(){
17       s=document.getElementById('start');
18       javaobjekt.startMp3();
19
20         if (s.innerHTML=='Pause')
21           s.innerHTML='Start';
22
23         else
24             s.innerHTML='Pause';
25       }
26
27     </script>
```

```
28    </head>
29
30
31    <body>
32    <ul>
33
34        <li id='start' onClick='pause();'>
35            Start
36        </li>
37
38        <li onclick="javaobjekt.stoppMp3()">
39           Stopp
40
41    </ul>
42    </body>
43
44 </html>
```

9.3 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 *servletbeholder (servlet/web container)* laster og starter servlets, håndterer hele dens livssyklus og kommunikasjon med webtjeneren. Ofte er web-tjener og serlvetbeholderen i samme programpakke.

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

9.3.1 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 servletbeholdere:
Apache Tomcat
GlassFish
Jetty

9.4 Servlet-beholderen Jetty

`http://www.eclipse.org/jetty/documentation/current/quickstart-running-jetty.htmlhttp://wiki.eclipse.org/Jetty/Starting`

- Skriptet under viser hvordan jetty kan installeres og startes.

Listing 9.3: eksempler/10/install-jetty.sh

```
1  #!/bin/bash
2
3  # Dette skriptet laster ned og installerer jetty i aktiv katalog
4  # av Thomas Nordli 19/4 2013
5
6  # Finn versjons nummer paa nettsiden og sett det inn under
7  #JETTY_VERSION=7.3.0.v20110203
8  JETTY_VERSION=7.6.10.v20130312
9
10
11 # Laster jetty ned hvis den ikke finnes fra foer
12 wget -c -O jetty-distribution-$JETTY_VERSION.tar.gz \
13     "http://eclipse.org/downloads/download.php?file=/jetty/stable-7/dist/
           jetty-distribution-$JETTY_VERSION.tar.gz&r=1"
14
15 if [ $? != 0 ]; then
16     echo Nedlasting feilet.
17     echo Avslutter installasjons-skript.
18     exit 1
19 fi
20
21 # Pakker opp
22 tar -xzf jetty-distribution-$JETTY_VERSION.tar.gz
23
24 if [ $? != 0 ]; then
25     echo Opppakking feilet.
26     echo Avslutter installasjons-skript.
27     exit 2
28 fi
29
30 ln -s jetty-distribution-$JETTY_VERSION jetty
31
32 echo jetty-distribution-$JETTY_VERSION er installert (med symlinken \'jetty
       \').
33 echo i katalogen $PWD/jetty-distribution-$VERSION
34 echo
35 echo
36 echo
37 echo For aa starte jetty:
38 echo ---------------------
39 echo $ cd jetty
40 echo $ java -jar start.jar
41 echo -------------------------
```

9.5 Eks: Hallo-verden

Listing 9.4: eksempler/10/ServletHallo.java

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

9.5.1 Kompliere til bytekode:

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

9.5.2 Katalogtre med filer:

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

Listing 9.5: eksempler/10/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>
```

9.6 war-filer

- Opprette web-archive:

jar -cf ServletHallo.war .

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

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

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

Listing 9.6: eksempler/10/mkservlet.sh

```
#!/bin/bash

# Dette skriptet lager et en enkel war-fil av en "servlet-java-fil"

if [ "$1" == "" ]; then
    echo Javafil maa oppgis som foerste argument.
    echo Javafilen maa ligge i aktiv katalog
    echo Flere filer som skal med kan nevnes som argument 2,3,.. etc.
    exit
fi

# Klassenavn er "fornavnet" til javafila
CLASS=$(basename $1 .java)

# katalognavn er klassenavn med smaa bokstaver
DIR=$(echo $CLASS | tr "A-Z" "a-z")

mkdir -p $DIR/WEB-INF/classes/

# Kompilerer
javac -d $DIR/WEB-INF/classes/ -cp .:/usr/share/java/servlet-api.jar $CLASS
    .java

# Lager web-xml-fila
cat <<EOF > $DIR/WEB-INF/web.xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://java.sun.com/
        xml/ns/j2ee/web-app_2_4.xsd"
    version="2.4">

  <servlet>
    <servlet-name>$CLASS</servlet-name>
    <servlet-class>$CLASS</servlet-class>
  </servlet>

  <servlet-mapping>
    <servlet-name>$CLASS</servlet-name>
    <url-pattern>/$CLASS</url-pattern>
  </servlet-mapping>

</web-app>
EOF

# Kopierer alle filene som skal med
cp $* $DIR/

# Lager war-fil i webapps-katalogen
```

```
48
49 cd $DIR
50 jar -cf $DIR.war .
51
52 echo \'Web archive\'-filen $DIR/$DIR.war er forsÃ¸kt opprettet.
```

9.7 Eksempel: Input fra html-skjema

9.7.1 Lag war-fil med kommando:

./mkservlet.sh SkjemaServlet.java skjemaservlet.html

9.7.2 Filer i eksemplet;

Listing 9.7: eksempler/10/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 9.8: eksempler/10/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 }
```

9.8 Eksempel: Informasjonskapsler

Listing 9.9: eksempler/10/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("Du␣har␣ingen␣kaker␣:(");
25
26          } else {
27              // skriver ut verdiene i informasjonskapslene
28              ut.printf("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  }
```

9.8.1 Lag war-fil med kommando:

./mkservlet.sh CookieTest.java

9.9 Eksempel: Sesjoner

Listing 9.10: eksempler/10/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  }
```

9.9.1 Lag war-fil med kommando:

./mkservlet.sh SesjonsTest.java

9.10 Øvelser

9.10.1 Oppgave 10.1

- 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.

9.10.2 Oppgave 10.2

- 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 10.1 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 10

JSP og JavaBeans

10.1 Løsningsforslag

10.1.1 Oppgave 10.1 (Tidligere 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 10.1: losninger/10/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 10.2: losninger/10/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  }
```

10.2 JSP

10.2.1 Hva er JSP?

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

Eksempel:

Listing 10.3: eksempler/11/jsp/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>
```

`http://oopva60.hive.no:8080/08/Hallo.jsp`

10.2.2 Hvorfor JSP ble introdusert?

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

- JSP er en annen måte å skrive servlets uten å være "java-ekspert"

I det opprinnelige JSP-API'et var det nødvendig med noe java-kode, men etterhvert er det utviklet et eget bibliotek med egne "tagger" - JavaServer Pages Standard Tag Library ( )

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

10.2.3 Når brukes JSP, og når brukes servlets?

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

Servlets

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

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

10.2.4 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"

10.2.5 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.
- 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>

(7. "Expression Language"-uttrykk)

ikke pensum

Et eksempel med kommentarer, direktiver og uttrykk:

Listing 10.4: eksempler/11/jsp/Klokka.jsp

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

10.2.6 Implisitte JSP-objekter

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

HttpServletRequest request

http-forespørsel fra klienten

HttpServletResponse response

http-responsen som skal sendes til klienten

HttpSession session

Sesjonsobjekt forbundet med request- og response- objektene

ServletContext application

har referanser til 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"

10.2.7 Tilgang til DB via JDBC - tre måter

1. plassere nødvendig kode i JSP'en

Listing 10.5: eksempler/11/jsp/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

tn@debbie:~/oopva60/jetty/webapps/08$ ls -l WEB-INF/lib/
totalt 0
lrwxrwxrwx 1 tn tn 40 2012-01-03 15:01 mysql-connector-java.jar -> /usr/share/java/mys
tn@debbie:~/oopva60/jetty/webapps/08$
```

2. definere spesialtilpassede "tags"

(ikke pensum)

3. bruke JavaBeans

vi ser eksempel på dette senere ...

10.2.8 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 10.6: eksempler/11/jsp/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 10.7: eksempler/11/jsp/Sesjonsdata.jsp

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

10.2.9 Error pages

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

eller bruke JSP-spesifisert måte å håndtere feil

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

i feilmeldings-JSP'en <%@ page isErrorPage="true" %>

Vi ser på et eksempel:

Listing 10.8: eksempler/11/jsp/Feilmelding.jsp

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


<HTML>
<HEAD>
  <TITLE>JSP-feilmeldings-side</TITLE>
<PRE>

JSP-feilmeldings-side
---------------------

Foelgende feil er oppstaatt:

<%=
exception.toString()
%>

</PRE>

</BODY>
</HTML>
```

Listing 10.9: eksempler/11/jsp/DBFailConnect.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://debbie.hive.no/bokbase",
  "dba",
  "da"
);

ResultSet res;
res = lnk.createStatement().executeQuery("show tables");
%>

<PRE>

<% while(res.next())
  out.println(res.getString(1));
%>

</PRE>
</BODY>
</HTML>
```

10.3 JavaBeans

10.3.1 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 metoder som skal være tilgjengelig utenfra bønna bør begynne med "get-" eller "set-" (eller "is-")
- bør implementere Serializable
- pakk eventuelt inn i jar-fil sammen med manifest-fil (påkrevd for bruk i IDE)

Listing 10.10: eksempler/11/beans/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 }
```

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

Minimal manifestfil (Manifest.mf):

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

10.3.2 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 "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'

Eks.:

- <jsp:setProperty name="halloVerdenBonne" property="hilsen" value="Hallo verden">
- <jsp:getProperty name="halloVerdenBonne" property="hilsen">

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`

10.3.3 Eksempler

Hallo

Listing 10.11: eksempler/11/beans/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 10.12: eksempler/11/beans/bonne.jsp

```
1  <html>
2  <head><title>Hallo-bonne test </title></head>
3  <body>
4
5     <jsp:useBean id="bonne" class="bonner.HalloBonne"/>
6
7     <%= bonne.getHilsen() %>
8
9     <jsp:getProperty name="bonne" property="hilsen" />
10
11    ${bonne.hilsen}
12
13
14
15 </body>
16 </html>
```

`http://dbada50.hive.no:8080/09/bonne.jsp`

Person 1

Listing 10.13: eksempler/11/beans/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 10.14: eksempler/11/beans/person1.jsp

```
1  <%@ page errorPage="Feilmelding.jsp" %>
2
3  <HTML>
4  <HEAD>
5    <TITLE> Person-skjema m/bonne </TITLE>
6  </HEAD>
7  <BODY>
8
9  <jsp:useBean id="pB" class="bonner.PersonBonne" />
10
11 <jsp:setProperty name="pB" property="fornavn" value="Thomas" />
12 <jsp:setProperty name="pB" property="etternavn" param="etternavn" />
13 <jsp:setProperty name="pB" property="lonn" />
14
15 <FORM method=post>
16  Fornavn
17  <INPUT
18    type=text
19    size=60
20    name=fornavn
21    value='<jsp:getProperty name="pB" property="fornavn" />'
22  /> <BR>
23
24  Etternavn
25  <INPUT
26    type=text
27    size=60
28    name=etternavn
29    value='<%= pB.getEtternavn() %>'
30  /> <BR>
31
32  Lonn
33  <INPUT
34    type=text
35    size=15
36    name=lonn
37    value='${pB.lonn}'
38  /> <BR>
39
40  <INPUT type=submit />
41 </FORM>
42
43 </BODY>
44 </HTML>
```

`http://dbada50.hive.no:8080/09/person1.jsp`

Person 2

Listing 10.15: eksempler/11/beans/person2.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="*" />
12
13 <FORM method=post>
14  Fornavn
15  <INPUT
16     type=text
17     size=60
18    name=fornavn
19     value='<jsp:getProperty name="pB" property="fornavn" />'
20  /> <BR>
21
22  Etternavn
23  <INPUT
24     type=text
25     size=60
26    name=etternavn
27     value='<%= pB.getEtternavn() %>'
28  /> <BR>
29
30  Lonn
31  <INPUT
32    type=text
33    size=15
34   name=lonn
35    value='${pB.lonn}'
36  /> <BR>
37
38  <INPUT type=submit />
39 </FORM>
40
41 </BODY>
42 </HTML>
```

`http://dbada50.hive.no:8080/09/person2.jsp`

JDBC

Listing 10.16: eksempler/11/beans/bonner/SqlBonne.java

```
1  package bonner;
2
3  public class SqlBonne implements java.io.Serializable {
4
5    private static java.sql.ResultSetMetaData met;
6    private static java.sql.Statement          stm;
7
```

```
8     private String sql = "show tables";
9
10    public SqlBonne() throws
11    java.sql.SQLException,
12    ClassNotFoundException {
13
14      Class.forName("com.mysql.jdbc.Driver");
15      stm = java.sql.DriverManager.getConnection(
16           "jdbc:mysql://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 10.17: eksempler/11/beans/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
```

```
    type=text
    size=100
    name=sporring value='<%= request.getParameter("sporring") %>'
  />
</FORM>

<jsp:useBean id="sqlBonne" class="bonner.SqlBonne" />
<jsp:setProperty name="sqlBonne" property="sporring" />
<jsp:getProperty name="sqlBonne" property="resultHtmlTable" />

</BODY>
</HTML>
```

`http://dbada50.hive.no:8080/09/sqlBonneKlient.jsp`

10.4 Øvelser

10.4.1 11.1

Løs oppgave 10.1 med JSP'er i stedet for som servlets.

10.4.2 11.2

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

10.4.3 11.3

a)

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

Listing 10.18: eksempler/11/jsp/SqlKlient2.java

```
import java.net.URLDecoder;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.sql.SQLException;
import java.sql.Statement;

public class SqlKlient2 {

  private static ResultSetMetaData met;
  private static Connection        lnk;
  private static Statement         stm;
  private static ResultSet         res;

  public static void main(String[] args)
  throws
  SQLException,
  ClassNotFoundException {

    String url, drv, usr, pwd, sql;

    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).

10.4.4 11.4

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.

10.4.5 11.5

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.

Del V

Obligatoriske oppgaver

Tillegg A

Obligatorisk oppgave 1 - 2012

Administrativt

- Besvarelsene innleveres i Fronter (1-2 pers. pr. besvarelse).
- Innleveringen pakkes pent inn i en enkelt fil. Utelat klassefilene.
- Innleveringen fremvises faglærer for godkjenning.
- Innleveringsfrist finnes i Fronter.

Tilbakemelding gjøres muntlig av faglærer. Eventuell godkjenning registreres av faglærer i Fronter. Studentene må selv ta initiativ til å vise besvarelsen til faglærer. Dersom besvarelsen underkjennes må ny og forbedret versjon leveres og vises for ny vurdering. Husk at alle de obligatoriske oppgavene må være godkjent ca. en uke før eksamen, for å kunne stille til eksamen.

Ikke nøl med å ta kontakt med faglærer for råd og veiledning underveis i arbeidet.

Oppgavetekst

Oppgaven skal løses med programmeringspråket Java og ved bruk av teknikker presentert i løpet av de fire første forelesningene.

a) Lag en klasse som representerer en unix-bruker. Ta med brukernavn og fullt navn som tekst-strenger, og UID som et heltall.

b) Lag en tcp-tjener til et unix/linux-system. Tjeneren skal

- kunne svare på tre ulike kommandoer: (1) *brukere*, (2) *enkeltbruker* og (3) *csv*.
- kunne betjene *flere klienter samtidig*.

(1) Dersom en klient sender kommandoen *brukere*, skal tjeneren levere en vektor bestående av en tabell med objekter av klassen du lagde i a). Objektene i tabellen skal representere alle brukere med UID $>$ 999. Forbindelsen til klienten kobles så ned.

(2) Ved mottak av kommandoen *enkeltbruker <brukernavn>* leveres et objekt av klassen du lagde i a). Objektet skal enten representere den forespurte brukeren, <brukernavn>, eller et "tomt" objekt som dersom brukeren ikke finnes på systemet (eller har UID < x1000). Forbindelsen til klienten skal *ikke* avsluttes etter at denne forespørselen er betjent.

(3) Når tjeneren får kommandoen *csv* fra en klient, skal den sende en kommaseparert fil til klienten som en vanlig strøm av tekst. Hver rad i fila skal representere en bruker med UID > 999. Forbindelsen til klienten kobles så ned.

c) Lag en testklient til tjeneren i b)

Tillegg B

Obligatorisk oppgave 2 - 2012

Administrativt

- Besvarelsene innleveres i Fronter (1-2 pers. pr. besvarelse).
- Innleveringen pakkes pent inn i en enkelt fil. Utelat klassefilene.
- Innleveringen fremvises faglærer for godkjenning.
- Innleveringsfrist finnes i Fronter.

Tilbakemelding gjøres muntlig av faglærer. Eventuell godkjenning registreres av faglærer i Fronter. Studentene må selv ta initiativ til å vise besvarelsen til faglærer. Dersom besvarelsen underkjennes må ny og forbedret versjon leveres og vises for ny vurdering. Husk at alle de obligatoriske oppgavene må være godkjent ca. en uke før eksamen, for å kunne stille til eksamen.

Ikke nøl med å ta kontakt med faglærer for råd og veiledning underveis i arbeidet.

Oppgavetekst

Bruk teknikker presentert i løpet av de *seks første forelesningene og øvingene.*

a) Lag en *http*-tjener til et unix/linux-system. Tjeneren skal

- implementeres med programmeringspråket Java.
- kunne svare på to ulike forespørsler:
  (1) GET /bruker HTTP/1.0
  (2) GET /bruker?<*BRUKER*> HTTP/1.0 [1]
- levere en webside med feilmelding, dersom annen en de to gyldige forespørslene kommer.
- kunne betjene *flere klienter samtidig.*

Ta gjerne utgangspunkt i tjener-koden du lagde i oblig1, men bruk en vanlig nettleser (f.eks. firefox) som testklient.

Her kommer en beskrivelse av hvordan tjeneren skal respondere på de to forespørslene:

[1] <BRUKER> er en tekst-streng som skal identifisere en bruker.

(1) Tjeneren skal levere et xml-dokument bestående av UID og fulltnavn for alle brukerne med UID > 999.

(2) Tjeneren skal levere et xml-dokument bestående av UID, og fulltnavn for brukeren/e som passer med tekststerengen <BRUKER> og har UID > 999.

Du kan se et eksempel på hvordan det skal virke på urlen: `http://oopva60.hive.no/oblig/brukere\_xml.cgi`. For å se hele responsen, inkludert http-hodet, kan følgende kommandoer kjøres:

```
1 AUTH=$(echo -n oop:va| base64)
2 echo -e ''GET http://oopva60.hive.no/oblig/brukere_xml.cgi HTTP/1.0\
    nAuthorization: Basic $AUTH\n\n''| nc debbie.hive.no 80
```

b) Lag en web-side ved, hjelp av HTML og JavaScript. Den skal gi brukeren et brukergrensesnitt med en trykk-knapp og et tekstfelt. Når trykknappen trykkes ned skal det sendes en forespørsel til tjeneren du lagde i a). Xml-dataene fra tjeneren skal presenteres som en html-tabell under trykk-knappen og tekstfeltet.

Du kan se et eksempel på hvordan brukergrensesnittet skal se ut etter at 'Nordli' er skrevet i tekstfeltet og kappen er trykket på følgende url: `http://oopva60.hive.no/oblig/gui-eksempel.html`.

Tillegg C

Obligatorisk oppgave 3 - 2012

Administrativt

- Bruk teknikker presentert i dette kurset til å løse oppgaven.
- Besvarelsene innleveres i Fronter (1-2 pers. pr. besvarelse).
- Innleveringen pakkes pent inn i en enkelt fil.
- *Ikke* levere klassefilene.
- Innleveringen fremvises faglærer for godkjenning.
- Innleveringsfrist finnes i Fronter.

Tilbakemelding gjøres muntlig av faglærer. Eventuell godkjenning registreres av faglærer i Fronter. Studentene må selv ta initiativ til å vise besvarelsen til faglærer. Dersom besvarelsen underkjennes må ny og forbedret versjon leveres og vises for ny vurdering. Husk at alle de obligatoriske oppgavene må være godkjent ca. en uke før eksamen, for å kunne stille til eksamen.

Ikke nøl med å ta kontakt med faglærer for råd og veiledning underveis i arbeidet.

Oppgavetekst

I denne oppgaven skal du lage en hybrid webapplikasjon for android-plattformen. Applikasjonen skal hente data fra web-tjeneren du lagde i forrige oblig og lagre det lokalt på android-enheten.

Utvid web-siden fra forrige oblig med en mulighet for å lagre kontakter i android-enhetens kontaktdatabase. Du kan se et eksempel på hvordan brukergrensesnittet skal se ut etter at 'Nordli' er skrevet i tekstfeltet og søkekappen er trykket på følgende url: `http://oopva60.hive.no/oblig/gui-eksempel2.html`. La denne utvidede web-siden være brukergrensesnittet i din hybrid-app.

For å lagre kontakter i android-enhetens kontaktdatabase *kan* dere bruke java funksjonen leggTilKontakt() som dere ser under. Funksjonen skal virke på android-versjoner f.o.m. 2.0. Husk at for å få tilgang til java-metoder fra javascript kan dere bruke et *Javascript-Interface* (som vist med eksempel i forelesningen *Android og Javascript*).

Husk å deklarere hvilke spesialrettigheter appen må ha i *AndriodManifest.xml*. Denne appen trenger rettigheter til å skrive i kontakt-databasen og hente data fra internett.

```
1
2  public void leggTilKontakt(String navn, String uid) {
3
4    // Ny kontakt
5
6    android.content.ContentValues values = new android.content.ContentValues
         ();
7    values.put(RawContacts.ACCOUNT_TYPE, (String)null );
8    values.put(RawContacts.ACCOUNT_NAME, (String)null );
9
10   android.net.Uri raaKontaktUri = getContentResolver().insert(RawContacts.
         CONTENT_URI, values);
11   long raaKontaktId = android.content.ContentUris.parseId(raaKontaktUri);
12
13
14   // Kontaktens navn
15
16   values.clear();
17   values.put(RawContacts.Data.RAW_CONTACT_ID, raaKontaktId);
18   values.put(RawContacts.Data.MIMETYPE, CommonDataKinds.StructuredName.
         CONTENT_ITEM_TYPE);
19   values.put(CommonDataKinds.StructuredName.DISPLAY_NAME, navn);
20   getContentResolver().insert(ContactsContract.Data.CONTENT_URI, values);
21
22   // Kontaktens "nick" (=UID)
23
24   values.clear();
25   values.put(RawContacts.Data.RAW_CONTACT_ID, raaKontaktId);
26   values.put(RawContacts.Data.MIMETYPE, CommonDataKinds.Nickname.
         CONTENT_ITEM_TYPE);
27   values.put(CommonDataKinds.Nickname.NAME, uid);
28
29   getContentResolver().insert(ContactsContract.Data.CONTENT_URI, values);
30
31 }
```

Del VI

Eksamensoppgaver

Tillegg D

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 E

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 F

Kontinuasjonseksamen for 2010-eksamen

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 og bønne (bean)?

a)

Hva er felles for de nevnte teknologiene?

b)

Forklar hva som særpreger dem.

Oppgave 2 (20%)

Skriv et java-program som sender filen *textfil.txt* v.h.a. transportprotokollen TCP. Inkluder forklarende kommentarer.

Oppgave 3 (20%)

a)

Forklar hvordan en kappløp-situasjon kan oppstå. Vær konkret. Skriv gjerne kode.

b)

Forklar og vis med kode-utdrag hvordan vi, med Java, kan hindre kappløp.

Oppgave 4 (10%)

Et web-basert program tar i mot input-parametre fra skjemaet:

```
1 <form action=X>
2  <input type=text name=Y>
3  <input type=text name=Z>
4  <input type=submit>
5 </form>
```

Gi en beskrivelse av dette skjemaet.

Oppgave 5 (30%)

Programmet nevnt i Oppgave 4, tar i mot input-parametrene fra skjemaet i oppgave 4 og returnerer dem i et tekstformat, som en kvittering tilbake til klienten.

Kvitteringen består av en linje for hvert skjema-element. Linjene er på følgende form:

```
navn = verdi
```

Hvordan ville du implementert dette programmet, som ...

a)

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

b)

en servlet. Vær konkret, skriv gjerne kode.

c)

en JSP. Vær konkret, skriv gjerne kode.

Tillegg G

Eksamen våren 2011 - uten 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%)

Tenk deg at du skal lage en applikasjon som henter noen parametre fra brukeren, gjør noen tunge beregninger og presenterer resultatet for brukerne. Kjøringen av applikasjonene kan tenkes fordelt på en eller flere datamaskiner. Applikasjonen skal implementeres ved hjelp av teknologi/teknikker, hentet fra pensumet du nå eksamineres i.

Gi en faglig begrunnelse for hvordan du ville implementert applikasjonen hvis du forutsetter at maskinen med brukergrensesnittet

a)

skal være en android-telefon *uten* garantert internett-tilkobling.

b)

skal være en laptop *med* konstant internett-tilkobling.

c)

kan være en laptop *eller* en android-telefon *med* konstant internett-tilkobling.

d)

kan være en laptop *eller* en android-telefon *uten* garantert internett-tilkobling.

Oppgave 2 (40%)

a) 20%

Programmér en tjener (server) som tar i mot et datagram v.h.a. transportprotokollen UDP. Inkludér forklarende kommentarer.

b) 10%

En tjener tar i mot datagrammer som i a). Den sender også kvitteringer tilbake til avsenderen. Dersom du skulle sørge for tjeneren kunne håndtere flere forespørsler uten å måtte vente på at forrige forespørsel var ferdig kvittert, har du to teknikker å velge mellom fra pensum. Forklar i korte trekk hvordan de to teknikkene virker, og hvordan de skiller seg fra hverandre. Skriv gjerne kode for å eksemplifisere.

c) 5%

Får vi en kappløps-situasjon i tjeneren fra a), dersom vi *ikke* implementerer teknikkene fra b)? Begrunn Svaret.

d) 5%

Får vi en kappløps-situasjon i tjeneren fra a), dersom vi implementerer teknikkene fra b)? Begrunn svaret.

Oppgave 3 (15%)

```
1  package xx.yy;
2
3  public class XxYy 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, Zyz.class);
14         wv = new android.webkit.WebView(this);
15         wv.loadUrl(``file:///android_asset/index.html'');
16         wv.getSettings().setJavaScriptEnabled(true);
17         wv.addJavascriptInterface( new Yyyyy(this), ``zzz'' );
18         setContentView(wv);
19     }
20
21     public class Yyyyy {
22
23       android.content.Context kontekst;
24       public Yyyyy(android.content.Context kontekst) { super(); }
25       public void start(){ startService(intent); }
```

```
26          public void stop() { stopService(intent); }
27          }
28  }
```

a) 10%

Gi en kort overordnet beskrivelse av hva koden gjør. Sørg spesielt for å få med hensikten med de to klassene og de fire metodene du finner i koden.

b) 5%

Gi en detaljert beskrivelse, linje for linje, av hva som skjer i når koden i linjene 13-18 kjøres.

Oppgave 4 (25%)

```
1  <html><head><title>x</title></head>
2    <body>
3      <form action=x>
4       <input type='text' name='x'>
5       <input type='text' name='y'>
6       <input type='text' name='z'>
7       <input type=submit>
8      </form>
9    </body>
10 </html>
```

a)

Gi en kort overordnet beskrivelse av koden

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.

c)

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

d)

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

e)

Oppgi (kortfattet) fordeler og ulemper ved å implementere skjema-behandlingen som en JSP.

Tillegg H

Eksamen våren 2012 - uten 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 (10%)

Under ser du koden til en java-klasse.

```
1  public class F{
2
3      public static void main(String[] args)
4          throws java.io.IOException {
5
6          java.io.ObjectOutputStream a;
7          java.io.RandomAccessFile   b;
8          java.io.FileOutputStream   c;
9
10         b = new java.io.RandomAccessFile("x","rw");
11         c = new java.io.FileOutputStream("y");
12         a = new java.io.ObjectOutputStream(c);
13
14         // ... mer kode ..
15     }
16 }
```

a og *b* brukes for å lagre data på forskjellige måter. Beskriv forskjellene.

Oppgave 2 (40%)

Under ser du koden til en *web-applikasjon.*

```
1  <!DOCTYPE html>
2  <html lang='no'>
3    <head>
4      <meta charset='utf-8'>
5      <title>Oppgave 2</title>
6      <script>
7      function t() {
```

```
8         a = document.getElementById('a').value
9         b = document.getElementById('b').value
10        c = document.getElementById('c')
11        c.value = a + b
12      }
13      </script>
14    </head>
15    <body>
16      <form action='javascript:t()'>
17        <input type='text' id='a'>
18        <input type='text' id='b'>
19        <input type='text' id='c'>
20        <input type='submit'>
21      </form>
22    </body>
23  </html>
```

a) Gi en detaljert beskrivelse av hva som skjer når koden lastes i en nettleser (web browser).

b) Beskriv kodelinje for kodelinje hvordan en brukersesjon vil forløpe.

c) Vis hvilke endringer du vil gjort for å tilpasse koden til en android-telefon. Begrunn svaret.

d) Tenk deg at en del av brukerne har nettlesere (web browsers) uten støtte for javascript. Vis hvordan du ved ville brukt metoder/teknologi fra pensum til å lage en versjon av web-applikasjonen som virker på slike nettlesere. Begrunn valg av metoder/teknologi.

Oppgave 3 (30%)

Under ser du koden til en del av en *android-applikasjon.*

```
1  package a.b;
2  public class Oppgave3 extends android.app.Activity {
3    android.webkit.WebView v;
4      @Override
5      public void onCreate(android.os.Bundle savedInstanceState)
6      {
7          super.onCreate(savedInstanceState);
8          v = new android.webkit.WebView(this);
9          v.loadUrl("file:///android_asset/x.html");
10         v.getSettings().setJavaScriptEnabled(true);
11         v.addJavascriptInterface(new Y(this),"z");
12         setContentView(v);
13     }
14 }
```

a) Hva forteller det deg at "Oppgave3 extends android.app.Activity"?

b) Beskriv hvor "x.html" er lagret og hvilken hensikt den har i denne applikasjonen.

c) På linje 11 finner du metoden *addJavascriptInterface()*. Det første argumentet er en referanse til et objekt av klassen Object. Det andre argumentet er en tekstkonstant. Forklar og vis med eksempel hvordan *objektet av klassen* Y og tekstkonstanten z kan brukes.

Oppgave 4 (20%)

Tenk deg at du jobber som programmerer og skal programmere en *tjener (server)* i java for en kunde. Tjeneren skal kunne kommunisere med *flere klienter samtidig* v.h.a. *TCP*. Du skal velge å implementere den med metoder som er *blokkerende* eller *ikke-blokkerende*.

a) For å velge mellom blokkerende eller ikke-blokkerende metoder trenger du å innhente informasjon fra kunden. Hva ville du spurt kunden om? Begrunn svaret.

b) Vis hvordan du ville skrevet en tjener v.h.a. blokkerende metoder.

Tillegg I

Eksamen våren 2013 - uten løsningsforslag

Forsøk å besvare oppgavene 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 (50%)

Under ser du innholdet av filen *EksamensEksempel.java*. Les gjennom filen og svar på spørsmålene.

Listing I.1: EksamensEksempel.java

```
1  public class EksamensEksempel  extends Thread {
2
3      private java.net.Socket s;
4
5      public static void main(String[] args) throws Exception {
6          java.net.ServerSocket ss = new java.net.ServerSocket(8888);
7          while (true)
8              new EksamensEksempel(ss.accept());
9      }
10
11     public EksamensEksempel(java.net.Socket socket)  {
12         this.s = socket;
13         this.start();
14     }
15
16     @Override
17     public void run() {
18         java.io.PrintWriter su = null;
19         java.util.Scanner si = null;
20         java.util.Scanner fi = null;
21         try {
22             su = new java.io.PrintWriter(s.getOutputStream());
23             si = new java.util.Scanner(s.getInputStream());
24             fi = new java.util.Scanner(
25                 new java.io.FileInputStream(
26                     si.nextLine().split(" ")[1].replaceFirst("/", "")));
27             while (fi.hasNextLine())
28                 su.println(fi.nextLine());
```

```
29              fi.close();
30              su.flush();
31          } catch (java.io.FileNotFoundException e) {
32              su.println("HTTP/1.1 404 Not Found");
33              su.println("Content-Type: text/plain; charset=UTF-8");
34              su.println("");
35              su.println("404 - Ikke funnet");
36              su.flush();
37          } catch (java.io.IOException e) {}
38          try {
39              s.close();
40          }
41          catch (java.io.IOException e) {}
42      }
43  }
```

a) Gi en overordnet beskrivelse av klassen *EksamensEksempel*.

b) Forklar hensikten med metoden *main()* og gi en detaljert beskrivelse av den.

c) Forklar hensikten med konstruktøren og gi en detaljert beskrivelse av den.

d) Forklar hensikten med metoden *run()* og gi en detaljert beskrivelse av den.

e) Forklar hva som menes med *ikke-blokkerende IO* og gi ett begrunnet svar på om det brukes i klassen *EksamensEksempel*.

Oppgave 2 (30%)

a) Tenk deg at en bruker har fylt ut et skjema på *http://debbie.hive.no/skjema.html* og trykt på send-knappen. Videre kan du tenke deg at brukerens nett-leser (web browser) sender en *http-forespørsel* (*http request*) som begynner slik:

```
GET /eksempel.jsp?fnavn=Tom&enavn&Nord HTTP1.1
Host: debbie.hive.no
```

Vis hvordan filen *skjema.html* kunne sett ut.

b) Vis hva du ville gjort for å sørge for at visningen av *skjema.html* fra a) ble automatisk tilpasset til ulike skjermstørrelser?

c) Klassen *android.webkit.WebView* har metoden *setJavaScriptInterface()*. Forklar hva metoden brukes til og vis med et eksempel hvordan javascriptgrensesnittet kunne vært brukt i *skjema.html* fra a).

Oppgave 3 (20%)

Listing I.2: EksamensEksempel2.java

```
1  public class EksamensEksempel2  extends javax.servlet.http.HttpServlet {
2
3      public void doGet(javax.servlet.http.HttpServletRequest  req,
4              javax.servlet.http.HttpServletResponse res )
5    throws Exception {
6
7          res.setContentType("text/plain");
8          java.io.PrintWriter ut = res.getWriter();
9          ut.println("Funnet!");
10         ut.flush();
11     }
12 }
```

a) Gi en overordnet beskrivelse av hva klassen EksamensEksempel2 gjør. Forklar også hva slags program som objekter av klassen er ment å inngå i.

b) Vis hvilke endringer du ville gjort i klassen for at det skulle fungert som et *CGI-program*. Forklar også hvilke fordeler og ulemper en slik endring vil medføre.