Løsning på småoppgaver etter hvert underkapittel, kapittel 2-12

Kun til bruk sammen med læreboka "Programmering i Java", Else Lervik og Vegard B. Havdal. Stiftelsen TISIP og Gyldendal Akademisk.

Tilpasset 4.utgave av boka.

Kapittel 2.1

Oppgave 1

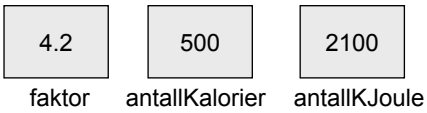
Følgende ord er lovlige navn:

I
dag
KarisBokevas_sykkel
AntallBær
NUMMER_Tall34

Oppgave 2

Reserverte ord: class, public, static, void, final, double

Variabler: faktor, antallKalorier, antallKJoule.



Kapittel 2.2

Oppgave 2

```
/**
 * Oppg2_2_2.java
 *
 * Programmet beregner areal og omkrets av et rektangel.
 */

class Oppg2_2_2 {
  public static void main(String[] args) {
    double lengde = 5.0;
    double bredde = 2.3;
    double arealet = lengde * bredde;
    System.out.println("Arealet av rektangelet er " + arealet + " kvadratmeter.");
    double omkretsen = lengde + lengde + bredde + bredde;
    System.out.println("Omkretsen av rektangelet er " + omkretsen + " meter.");
  }
}

/* Kjøring av programmet:
Arealet av rektangelet er 11.5 kvadratmeter.
Omkretsen av rektangelet er 14.600000000000001 meter.
```

```
*/
```

Kapittel 2.4

Oppgave 1

Ordlyden i feilmeldingene kan variere noe mellom de ulike versjonene av Java-kompilatoren. Feilmeldingene nedenfor kommer fra *java version 1.6.0_07*.

Feilmelding på linjen

```
final int ANTALL_ÅR = 35
```

"';' expected."

Semikolon mangler i *slutten* av linjen - og ikke der markøren peker (under _).

Feilmelding på linjen

```
double Beløp = 40,95;
```

"<identifier> expected."

Kompilatoren sier at den forventer navnet på noe. Den peker på siste bokstav i Beløp. Problemet her er faktisk desimalkommaet foran 9-tallet. Husk at Java krever punktum som desimalskille. Dette er et eksempel på at kompilatoren ikke alltid gir informative feilmeldinger. Setningen skal se slik ut:

```
double Beløp = 40,95;
```

Feilmelding på linjen

```
final EnKonstant = 789.6;
```

"<identifier> expected."

Kompilatoren forventer også her navnet på noe, og i dette tilfellet er dette tilfelle. En datatype mangler. Setningen skal se slik ut:

```
final double EnKonstant = 789.6;
```

Da får vi to feilmeldinger, den første på linjen

```
DOUBLE tall = 15.7;
```

"cannot find symbol, symbol: class DOUBLE."

Ettersom DOUBLE ikke er en primitiv datatype (akkurat det er det ikke lett for deg å vite!) antar Java at det er en klasse. Og Java finner ingen klasse med dette navnet. Vi går ut fra at det er double med små bokstaver som menes her:

```
double tall = 15.7;
```

Neste setning:

```
Beløp = Beløøp * 100;
```

gir meldingen "cannot find symbol, symbol: variable Beløøp". Her har vi antakelig en skrivefeil. Ordet Beløøp er ikke deklarert. Vi mener nok å skrive Beløp.

Alle setningene skal dermed se slik ut:

```
final int ANTALL_ÅR = 35;
double Beløp = 40.95;
final double EnKonstant = 789.6;
double tall = 15.7;
```

```
Beløp = Beløp * 100;
```

Oppgave 2

Beløp: Navn på variabler skal ha liten forbokstav.

EnKonstant: Konstanter skal skrives med store bokstaver og _, slik EN_KONSTANT.

Kapittel 2.5

Oppgave 1

12.45	double
"Hallo!"	String
"\""	String
'\"'	char
12345L	long
0456	int (i 8-tall-systemet, se fotnote 5 side 60)
true	boolean
3.245e-5	double

Oppgave 2

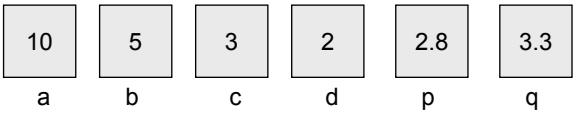
a) prisen - double
b) varemengde målt i kilo - double
c) varemengde målt i antall - int
d) fargekode (bokstav) - char
e) strekkode - String
f) navnet - String

Oppgave 3

```
Anne Eliassen Jensen
Tallet er 0.023
Tegnet er a
```

Kapittel 2.7

Oppgave 1



Oppgave 2

Følgende to setninger er lovlige:

```
b = b + c;
d = d;
```

Oppgave 3

c + d * a, verdi 23
a * b / c + a, verdi 26

```
c % d, verdi 1
d % c, verdi 2
p % q, verdi 2.8
q % p, verdi 0.5
c % d % a + b / c, verdi 2
a = b = 16, verdi 16
```

Oppgave 4

```
b * b - 4 * a * c
x * x + y * y + z * z
x * x * x
(a - b) * (a + b)
(a + b) / (c + d)
```

Merk at Java, i motsetning til en del andre programmeringsspråk, ikke har egen operator for potensering.

Kapittel 2.8

Oppgave 1

```
p + (double) a / q, verdi 5.83
p + a / q, verdi 5.83
(int) p + (int) q, verdi 5
(int) (p + q), verdi 6
```

Oppgave 2

Setningen a = p + q; krever casting på grunn av at vi ved tilordningen har omforming til en mindre datatype (fra double til int). For at setningen skal kunne utføres må den se slik ut: a = (int) (p + q); Innholdet i variablene a, b og d etter at de fire setningene er utført er som følger: a = 10, b = 0, d = 0.

Oppgave 3

```
a = -129542144, b = 60000000000
```

Vi ser at a er beregnet feil. Det skyldes at resultatet regnes ut som et uttrykk av datatypen int, og vi overskrider tallområdet for denne dataverdien. (Omformingen til long skjer først etter at beregningen er foretatt.) b blir riktig fordi tallområdet til høyre side nå er long.

Kapittel 3.1

Oppgave 1

Linje 12–13:

```
double tall1 = Double.parseDouble(tall1Lest);
double tall2 = Double.parseDouble(tall2Lest);
```

Linje 17:

```
double svar = tall2 - tall1;
```

Linje 20:

```
double svar = tall1 - tall2;
```

Kapittel 3.3

Merk at det dessverre er sneket seg inn en feil i nummereringen av oppgavene. Linjen under kodebiten ("Hva er verdien til ... ") hører til oppgave 1.

Oppgave 1

sum er lik 10.

Oppgave 2a

I alt fem variabler deklareres:

I blokk 1: tall1 og tall2
I blokk 2: tall3 og tall4
I blokk 3: tall3

To av variablene har samme navn.

Oppgave 2b

Deklarasjonene i blokk 1:

Skopet til tall1 er linje 2-17.
Skopet til tall2 er linje 3-17.

Deklarasjonene i blokk 2:

Skopet til tall3 er linje 6-12.
Skopet til tall4 er linje 9-12.

Deklarasjonen i blokk 3:

Skopet til tall3 er linje 13-15.

Oppgave 2c

```
tall1 = 60, tall2 = 50
tall1 = 30, tall2 = 100
tall3 = 20, tall4 = 150
tall1 = 30, tall2 = 100
```

Oppgave 2d

```
tall1 = 60, tall2 = 50
tall3 = 65
tall1 = 60, tall2 = 50
```

Kapittel 3.3

Oppgave 1a

```
if (antall > 20) {
  kode = 'M';
} else {
  kode = 'F';
}
```

Oppgave 1b

```
if (vekt / (høyde * høyde) > 25) {
  System.out.println("Du er for ting");
}
```

Oppgave 2

Hvis størrelse er større enn 38 skjer ingenting. I motsatt fall skrives teksten Liten! ut på skjermen. Kodebiten bør skrives som følger:

```
if (størrelse <= 38) {
  System.out.println("Liten!");
}
```

Oppgave 3

```
int a = 20;
int b = 30;
int c = 40;
if (a > b) {
  a = b;
} else {
  a = c;
  b = 50;
  if (a > 50) {
    a = 100;
  }
}
System.out.println("a = " + a + ", b = " + b + ", c = " + c);
```

Utskrift:

```
a = 40, b = 50, c = 40
```

Oppgave 4

Anbefalt kode fra og med første if-setning ser slik ut:

```
if (a < b) {
  a = b;
}
b = 10;
if (p == 20) {
  q = 13;
} else {
  q = 17;
}
if (r > s) {
  q = 100;
}
s = 200;
```

Variablene har følgende verdier etter at kodebiten er kjørt:

a = 30
b = 10
p = 20
q = 100
r = 30
s = 200

Kapittel 3.4

Oppgave 1

Fra linje 15 og utover:

```
String melding;
if (poeng < 0) {
  melding = "Poengsummen kan ikke være negativ.";
} else if (poeng < 36) {
  melding = "Karakteren er F.";
} else if (poeng < 55) {
  melding = "Karakteren er E.";
} else if (poeng < 71) {
  melding = "Karakteren er D.";
} else if (poeng < 86) {
  melding = "Karakteren er C.";
} else if (poeng < 96) {
  melding = "Karakteren er B.";
} else if (poeng < 101) {
  melding = "Karakteren er A.";
} else {
  melding = "For stor poengsum, maks. " + MAKS + ".";
}
showMessageDialog(null, melding);
```

Kapittel 3.5

Oppgave 1

Syntaksfeil: Det er ikke tillatt å ramse opp flere verdier i samme case-etikett.

Logisk feil: break må legges inn til slutt under de case-etikettene der programkontrollen skal hoppe ut av switch-blokken.

Følgende programkode er antaglig mer i samsvar med det programmereren har ment:

```
switch (ukedag) {
case 1:
  /* ikke break */
case 2:
  System.out.println("Begynnelsen av uka");
  break;
case 3:
  /* ikke break */
case 4:
  System.out.println("Midt i uka");
  break;
case 5:
  System.out.println("Slutten av uka");
  break;
case 6:
  /* ikke break */
case 7:
  System.out.println("Helg");
```

```
    break;
  default:
    System.out.println("Ugyldig ukedag: " + ukedag);
    break;
  }
```

Kodebiten i oppgaven inneholder også et semikolon helt til slutt. Dette er unødvendig.

Kapittel 3.7

Oppgave 1

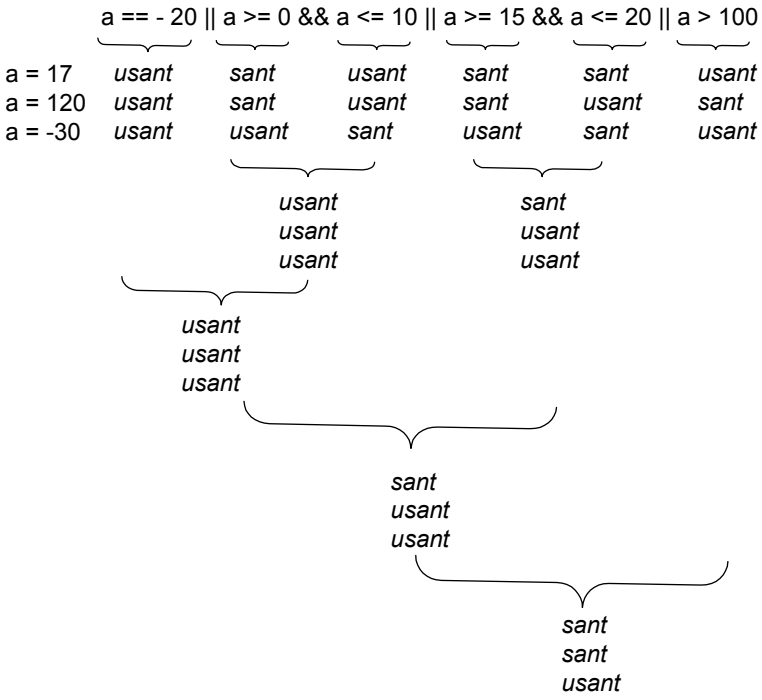
a) true
b) false
c) false
d) true

Oppgave 2

```
a) antallElever > 20 && antallElever < 30;
b) loddNr == 3 || loddNr == 18 || loddNr == 25;
c) svar == 'j' || svar == 'J';
d) temp < 15 || temp > 25;
e) sum > 0 && sum <= 10 || sum >= 100;
f)  tegn >= 'A' && tegn <= 'Z' || tegn >= 'a' && tegn <= 'z' ||
      tegn == 'æ' || tegn == 'ø' || tegn == 'å' || tegn == 'Æ' || tegn == 'Ø' || tegn == 'Å';
g) tegn >= '0' && tegn <= '9';
```

Oppgave 3

a = 17 og a = 120 gir at verdien til uttrykket er sant, mens a = -30 gir verdien usann.



Kapittel 4.1

Oppgave 1

Null ganger:

```
int teller = 0;
while (teller < 0) {
  System.out.println("Dette er en linje");
  teller = teller + 1;
}
```

Uendelig mange ganger (vi "kommenterer bort" oppdatering av telleren):

```
System.out.println("***Uendelig mange ganger:");
teller = 0;
while (teller < 5) {
  System.out.println("Dette er en linje");
  // teller = teller + 1;
}
```

Oppgave 2

```
import static javax.swing.JOptionPane.*;
class Oppg4_1_2 {
  public static void main(String[] args) {
    String antLinjerLest = showInputDialog("Antall linjer: ");
```

```
    int antLinjer = Integer.parseInt(antLinjerLest);
    int teller = 0;
    while (teller < antLinjer) {
      System.out.println("Dette er en linje");
      teller = teller +1;
    }
  }
}
```

Kapittel 4.2

Oppgave 1

2
2

Oppgave 2

a = 121
b = 20
c = 1

Kapittel 4.3

Oppgave 1

Velger å samle all tekst som skal skrives ut i variabelen melding. Skriver kun ut summen (og gjennomsnittet) dersom minst ett tall er lest inn.

```
import static javax.swing.JOptionPane.*;
class Oppg4_3_1 {
  public static void main(String[] args) {
    int antall = 0;
    int sum = 0;
    String tallLest = showInputDialog("Skriv tall, avslutt med Esc.");
    while (tallLest != null) {
      int tall = Integer.parseInt(tallLest);
      sum += tall;
      antall++;
      tallLest = showInputDialog("Skriv tall, avslutt med Esc.");
    }
    String melding = "Antall tall lest er " + antall + ".";
    if (antall == 0) {
      melding += "\nKan ikke beregne gjennomsnittet.";
    } else {
      double snitt = (double) sum / (double) antall;
      melding += "\nGjennomsnittet er " + snitt + ", og ";
      melding += "\nsummen er " + sum + ".";
    }
    showMessageDialog(null, melding);
  }
}
```

Oppgave 2

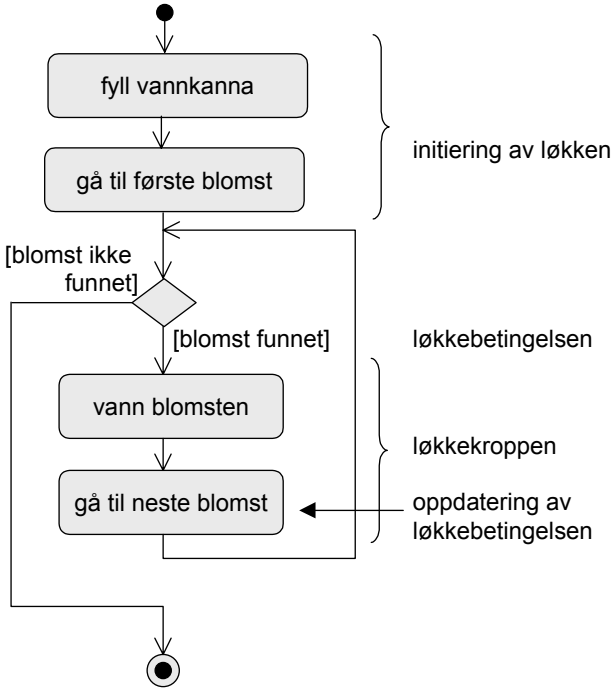
```
import static javax.swing.JOptionPane.*;
class Oppg4_3_2 {
  public static void main(String[] args) {
    String lestGrunntall = showInputDialog("Skriv grunntallet: ");
    String lestEksponent = showInputDialog("Skriv eksponenten: ");
    double x = Double.parseDouble(lestGrunntall);
    int n = Integer.parseInt(lestEksponent);
    double svar = 1;  // et tall opphøyd i 0 er lik 1, ikke 0.
    int teller = 0;  // telleren må starte på 0, hvis vi ikke skal endre løkkebetingelsen
    while (teller < n) {
      svar *= x; // svaret skal ganges med x, ikke med n
      teller++; // telleren skal økes, ikke n
    }
    showMessageDialog(null, "Svaret er " + svar);
  }
}
```

Oppgave 3

Programmet fungerer ikke for negativ eksponent.

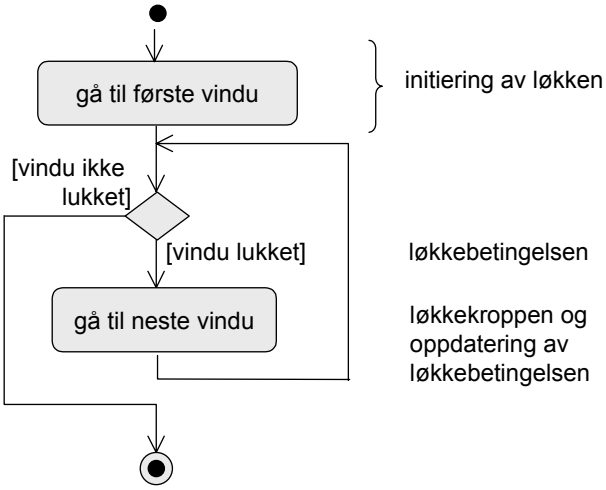
datasett nr.	n	forv. svar med grunntall -2	forv. svar med grunntall 0	forv. svar med grunntall +2
1	0	1	1	1
2	1	-2	0	2
3	3	-8	0	8

Oppgave 4a

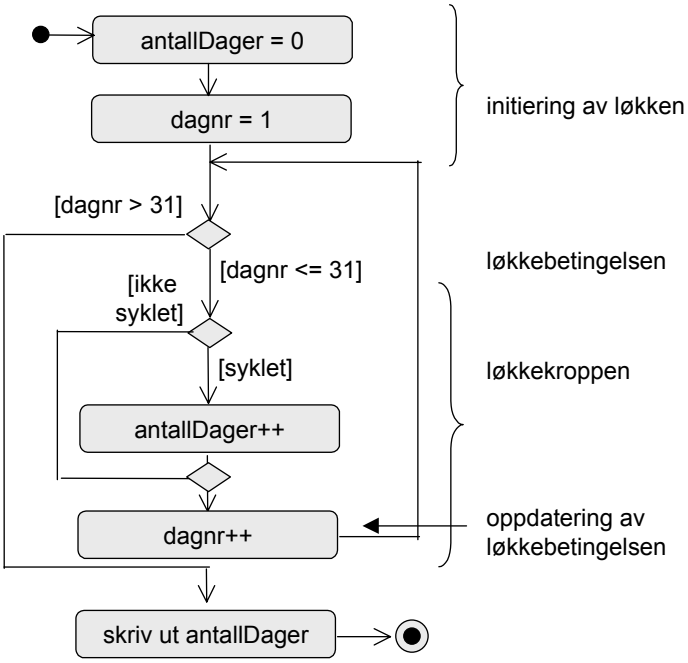


Vi forutsetter at det er nok å fylle vannkanna en gang. Hvis det ikke er det, får vi en test på om det er nok vann inne i løkka. Denne testen må ligge foran aktiviteten "vann blomsten". Hvis det ikke er nok vann, må vi fylle kanna før vi vanner.

Oppgave 4b



Oppgave 4c



Kapittel 4.4

Oppgave 1

```
class Oppg4_4_1 {
  public static void main(String[] args) {
    int antallTall = 0;  // ny variabel
    int antallTallSummert = 0;
    int sum = 0;
    int sumAlleTall = 0; // ny variabel
    int tall = 0; // deklarerer denne utenfor løkken nå
    String forrigeTallLest = "";
    String tallLest = showInputDialog("Skriv ett tall av gangen, avslutt med Esc: ");

    /* Gå i løkke så lenge som Esc ikke er tastet */
    while (tallLest != null) {
      if (!tallLest.equals(forrigeTallLest)) {
        tall = Integer.parseInt(tallLest);  // tall skifter verdi
        sum += tall;
        antallTallSummert++;
      }
      antallTall++;
      sumAlleTall += tall;
      forrigeTallLest = tallLest; // tar vare på siste innleste tall
```

```
      tallLest = showInputDialog("Skriv ett tall av gangen, avslutt med Esc: ");
    }
    showMessageDialog(null, "Summen er " + sum
                  + ". Vi har summert " + antallTallSummert + " tall.");
    if (antallTallSummert > 0) {  // gjennomsnittsverdiene oppgaven spør etter
      double snitt1 = (double) sum / (double) antallTallSummert;
      double snitt2 = (double) sumAlleTall / (double) antallTall;
      showMessageDialog(null, "Summerte tall, gjennomsnitt " + snitt1
                  + ",\n alle tall, gjennomsnitt " + snitt2 + ".");
    }
   }
  }
```

Kapittel 4.5

Oppgave 1

Feilen er at variabelen dag ikke endrer verdi. Vi får derfor en evig løkke.

Som for-løkke kan dette se slik ut:

```
for (int dag = 1; dag < 5; dag++) {
 System.out.println("Vakkert vær på dag nr " + dag);
}
```

Oppgave 2

Vi bruker nå en heltallsvariabel som løkketeller:

```
for (int grader = 0; grader <= 90; grader += 10) {
  double vinkel = Math.toRadians(grader);
  System.out.println("Sinus til vinkel " + grader + " grader er " + Math.sin(vinkel));
}
```

Kapittel 4.6

Oppgave 1

Kompileringsfeil:

```
cannot find symbol, symbol  : variable ord
```

i linjen

```
} while (ord != null);
```

Variabelen ord må være deklarert foran løkken:

```
String ord = "";
do {
```

I tillegg inneholder programmet en logisk feil. ord får verdien null når brukeren taster Esc for å avslutte. Denne verdien blir med i setningen som lages. Utskriften blir for eksempel

```
dette er noen ord null
```

Følgende kodeendring løser dette problemet:

```
String setning = "";
String ord = "";
```

```
do {
  setning += (ord + " ");
  ord = showInputDialog("Skriv ett ord, avslutt med Esc.");
} while (ord != null);
showMessageDialog(null, setning);
```

Vi får et blankt tegn i begynnelsen av setning, men det vises ikke ved utskrift i meldingsboksen. Det er mulig å fjerne dette tegnet ved å bruke String-metoder, se kapittel 8. Det mest elegante er nok å bruke en while-løkke her ... se oppgave 2, kapittel 4.7.

Kapittel 4.7

Oppgave 1

```
public static void main(String[] args) {
  int sum = 0;
  String tallLest = "0";
  do {
    int tall = Integer.parseInt(tallLest);
    sum += tall;
    tallLest = showInputDialog("Skriv tall, avslutt med Esc.");
  } while (tallLest != null);
  showMessageDialog(null, "Summen er " + sum + ".");
}
```

Selv om denne løsningen fungerer også dersom ingen tall leses inn, er det en dårligere løsning enn originalen. Måten vi håndterer ingen tall på er kunstig ved at vi starter med å sette tallLest lik "0". Det er bedre å ikke løpe gjennom løkken i det hele tatt i dette tilfellet. (Prøv begge variantene ved å multiplisere tallene i stedet for å legge dem sammen.)

Oppgave 2

Oppgaven på side 132 bør løses med en while-setning:

```
String setning = "";
String ord = showInputDialog("Skriv ett ord, avslutt med Esc.");
while (ord != null) {
  setning += (ord + " ");
  ord = showInputDialog("Skriv ett ord, avslutt med Esc.");
}
showMessageDialog(null, setning);
```

Kapittel 4.9

Oppgave 1

```
public static void main(String[] args) {
  int sum = 0;
  for (int linjeteller = 0; linjeteller < 10; linjeteller++) {
    sum += linjeteller;
    for (int tall = 1; tall <= linjeteller; tall++) {
      System.out.print(tall + " ");
    }
    System.out.println("Summen blir: " + sum);
```

```
    }
  }
```

Oppgave 2

Velger å spesialbehandle 0 før vi går inn i løkka:

```
  public static void main(String[] args) {
    System.out.println("Summen blir: 0, og produktet blir: 0.");
    int sum = 0;
    int produkt = 1;
    for (int linjeteller = 1; linjeteller < 10; linjeteller++) {
      produkt *= linjeteller;
      sum += linjeteller;
      for (int tall = 1; tall <= linjeteller; tall++) {
        System.out.print(tall + " ");
      }
      System.out.println("Summen blir: " + sum + ", og produktet blir: " + produkt + ".");
    }
  }
```

Oppgave 3

Velger å spesialbehandle 0 og 1 før vi går inn i løkka:

```
  public static void main(String[] args) {
    System.out.println("Summen av partallene blir: 0, og produktet blir: 0.");
    System.out.println("1 Summen av partallene blir: 0, og produktet blir: 0.");
    int sum = 0;
    int produkt = 1;
    boolean partall = true; // første tall er partall
    for (int linjeteller = 2; linjeteller < 10; linjeteller++) {
      if (partall) {
        produkt *= linjeteller;
        sum += linjeteller;
      }
      for (int tall = 1; tall <= linjeteller; tall++) {
        System.out.print(tall + " ");
      }
      System.out.println("Summen av partallene blir: " + sum
                         + ", og produktet blir: " + produkt + ".");
      partall = !partall; // annethvert tall er partall
    }
  }
```

Kapittel 4.10

Oppgave 1

Linje 28–29 skiftes ut med:

```
   /* Skriver ut resultatet */
  String melding = "Summen er " + sum;
  melding += (tallLest == null) ? ", brukeren har tastet Esc-tasten."
                                : ", grensen (" + GRENSE + ") er overskredet.";
  showMessageDialog(null, melding);
```

Kapittel 5.1 side 151

Linje 13–15 skiftes ut med:

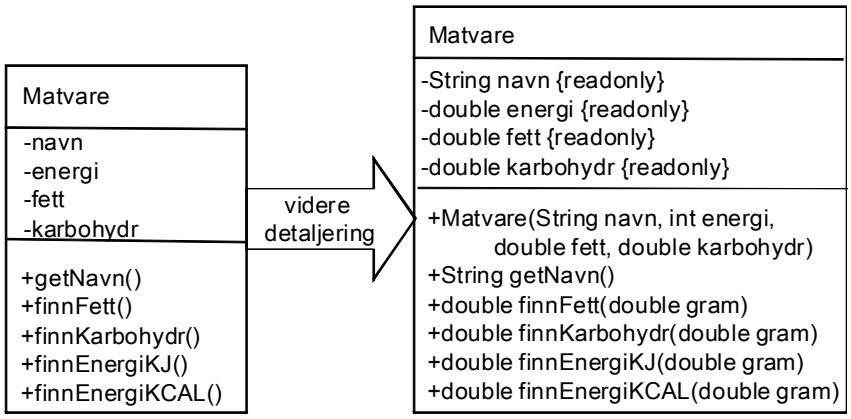
```
String beløpLest = showInputDialog("Oppgi transaksjon, avslutt med Esc: ");
while (beløpLest != null ) {
  double transaksjon = Double.parseDouble(beløpLest);
  olesKonto.utførTransaksjon(transaksjon);
  double saldo = olesKonto.getSaldo();
  System.out.printf("Ny saldo: kr %.2f.\n", saldo);
  if (saldo < 0.0) {
    showMessageDialog(null, "Negativ saldo!");
  }
  beløpLest = showInputDialog("Oppgi transaksjon, avslutt med Esc: ");
}
```

Først i koden må du legge inn setningen:

```
import static javax.swing.JOptionPane.*;
```

og klassen Konto (filen *Konto.java* fra eksempelsamlingen) må ligge i samme mappe som der du har løsningen til oppgaven.

Kapittel 5.1 side 156–157




```
/**
 *
 * Klassen Matvare.
 *
 * Et objekt av klassen Matvare vil inneholde informasjon om
 * næringsinnholdet i en matvare. Klassen kan brukes til å beregne
 * dette for et bestemt antall gram av varen.
 */

import static javax.swing.JOptionPane.*;
class Matvare {
  private final String navn;
  private final int energi;
```

```
  private final double fett;
  private final double karbohydr;

  /**
   * Konstruktøren tar data for 100 gram av varen.
   */
  public Matvare(String navn, int energi, double fett, double karbohydr) {
    this.navn = navn;
    this.energi = energi;
    this.fett = fett;
    this.karbohydr = karbohydr;
  }

  public String getNavn() {
    return navn;
  }

  public double finnFett(double gram) {
    return fett * gram * 0.01;
  }

  public double finnKarbohydr(double gram) {
    return karbohydr * gram * 0.01;
  }

  public double finnEnergiKJ(double gram) {
    return energi * gram * 0.01;
  }

  public double finnEnergiKCAL(double gram) {
    return energi * gram * 0.01 * 4.18;
  }
}

/**
 *
 * Klientprogram som beregner næringsinnhold for ulike mengder av
 * tre forskjellige matvarer.
 */
class Oppg5_1_side156 {
  public static void main(String[] args) {
    Matvare pomFritGkj = new Matvare("pommes frites, gatekjk.", 1381, 3.0, 39.8);
    Matvare pomFritFryst = new Matvare("pommes frites, fryst", 641, 2.2, 22.0);
    Matvare koktPotet = new Matvare("kokt potet", 339, 1.9, 17.1);

    String vektLest = showInputDialog("Vekt i gram, avslutt med Esc: ");
    while (vektLest != null ) {
      double vekt = Double.parseDouble(vektLest);

      System.out.print(pomFritGkj.getNavn());
```

```
            System.out.printf(": fett: %.1f, karbohydrater: %.1f, kJ: %.0f, kcal: %.0f.\n",
                pomFritGkj.finnFett(vekt), pomFritGkj.finnKarbohydr(vekt),
                pomFritGkj.finnEnergiKJ(vekt), pomFritGkj.finnEnergiKCAL(vekt));
            System.out.print(pomFritFryst.getNavn());
            System.out.printf(": fett: %.1f, karbohydrater: %.1f, kJ: %.0f, kcal: %.0f.\n",
                pomFritFryst.finnFett(vekt), pomFritFryst.finnKarbohydr(vekt),
                pomFritFryst.finnEnergiKJ(vekt), pomFritFryst.finnEnergiKCAL(vekt));
            System.out.print(koktPotet.getNavn());
            System.out.printf(": fett: %.1f, karbohydrater: %.1f, kJ: %.0f, kcal: %.0f.\n",
                koktPotet.finnFett(vekt), koktPotet.finnKarbohydr(vekt),
                koktPotet.finnEnergiKJ(vekt), koktPotet.finnEnergiKCAL(vekt));

            vektLest = showInputDialog("Vekt i gram, avslutt med Esc: ");
        }
      }
    }
```

Utskriftsetningene bruker System.out.printf() noe mer avansert enn vi har sett hittil. Eksempel på utskrift:

```
pommes frites, gatekjk.: fett: 6,0, karbohydrater:
79,6, kJ: 2762, kcal: 11545.

pommes frites, fryst: fett: 4,4, karbohydrater: 44,0,
kJ: 1282, kcal: 5359.

kokt potet: fett: 3,8, karbohydrater: 34,2, kJ: 678,
kcal: 2834.
```

Navnet på matvaren skrives ut med System.out.print(). Deretter brukes System.out.printf() for å få ut de fire desimaltallene med henholdsvis 1, 1, 0 og 0 desimaler. Formatstrengen har ett %-uttrykk for hvert av de fire tallene:

```
": fett: %.1f, karbohydrater: %.1f, kJ: %.0f, kcal: %.0f.\n"
```

Verdiene som skal skrives ut i henhold til dette formatet finner vi i rekke og rad etter formatstrengen, for eksempel:

```
pomFritGkj.finnFett(vekt), pomFritGkj.finnKarbohydr(vekt),
pomFritGkj.finnEnergiKJ(vekt), pomFritGkj.finnEnergiKCAL(vekt)
```

Kapittel 5.2

Oppgave 1

```
Konto arnesKonto = new Konto(123456789677L, "Arne Jensen", 1000);
```

Denne setningen inneholdt tre feil. To av dem er markert med fete typer. I tillegg var det et argument for mye i argumentlisten.

```
Konto johansKonto = new Konto(123456789677L, "Johan Hansen", 1000);
```

Vi må alltid ha et klassenavn etter new.

```
long kNr = arnesKonto.getKontonr();
```

Setningen slik den stod var ikke feil, men den hadde ingen hensikt. Enten må man lagre den verdien som hentes ut fra objektet i en variabel, eller man må bruke verdien i et uttrykk.

```
double saldo = arnesKonto.getSaldo();
```

Metoden getSaldo() tar ingen argumenter.

```
arnesKonto.utførTransaksjon(1000);
```

Metoden utførTransaksjon() har void som returtype. Vi får derfor ikke noen verdi vi kan ta i mot og lagre.

```
arnesKonto.utførTransaksjon(800);
```

Vi må sette opp navnet på et objekt, og ikke en klasse når vi skal sende en melding.

Oppgave 2

a) Per Ås er en bestemt person, og modelleres derfor som et objekt. En klasse er en beskrivelse av en mengde objekter med de samme attributtene og operasjonene.

b) Setningen utgjør et metodehode. Parentesen skal være tom eller inneholde parametere, det vil si beskrivelser av argumentene. "Januar" er et argument. Beskrivelsen vil være for eksempel String måned. Metoden bør da returnere lønnen for den oppgitte måneden. Det vil si at void må skiftes ut med en passende datatype, sannsynligvis double. Hittil har vi også hatt modifikatoren public foran. Dette kan utelates. Hva det innebærer, skal vi se på senere. Metodehodet kan for eksempel se slik ut: public double getLønn(String måned).

Ettersom metodenavnet begynner med get- kan dette også tolkes slik at en skal hente ut verdien til attributtet lønn (uavhengig av måned), og dermed bør metodehodet se slik ut: public double getLønn().

c) Samme her. 1756 må være et argument. Parameteren vil være for eksempel int årstall.

d) Her blandes begrepene. Vi kan si at "Objektet person har et attributt, navn."

e) Dette er et metodehode, se oppgave b) foran.

f) Setningen kan ikke omskrives til noe som det er mening i. Parameterne finner vi i metodehodet. Argumentene er de verdiene vi sender med metoden når vi bruker den for å sende en melding til et objekt. Vi sender også med argumenter til konstruktøren. Eksempel:

```
class Person {
  .....
  public double finnLønn(String måned)       // måned er en parameter
}
Person enPerson = new Person("Ole");  //  "Ole" er argument
.....
double lønnen = enPerson.finnLønn("Januar");  // "Januar" er argument
```

g) Attributter har ikke hode og kropp. En metode har hode og kropp. En klasse har hode og kropp. Et attributt er en egenskap ved et objekt. En bil har for eksempel attributtene merke, modell, registreringsnummer og farge.

h) En lokal variabel kan bare nås inne i den blokken der den er deklarert. En objektvariabel, derimot, kan nås i alle metoder i den klassen der den er deklarert.

Oppgave 3

```
class Sirkel {
  final private double radius;  // datatype mangler
  public Sirkel(double radius)  { // stor S
    this.radius = radius;  // this må være på venstre side av =
  } // sluttklamme mangler

  public double beregnAreal() { // arealet bør være datatypen double
    return Math.PI * radius * radius;
  }

  public double beregnOmkrets() {
    double omkrets = 2.0 * Math.PI * radius;  // må deklarere omkrets
    return omkrets;
  }
}
```

Oppgave 4

```
class Sirkelberegning{
  public static void main(String[] args) {
    Sirkel enSirkel = new Sirkel(20);
    double arealet = enSirkel.beregnAreal();
    System.out.println("Arealet er lik " + arealet);
    double omkretsen = enSirkel.beregnOmkrets();
    System.out.println("Omkretsen er lik " + omkretsen);
  }
}
```

Oppgave 5

Lokale variabler: Ingen.

Objektvariabler: kontonr, navn, saldo.

parametere:

I konstruktøren: kontonr, navn, saldo

I utførTransaksjon(): beløp

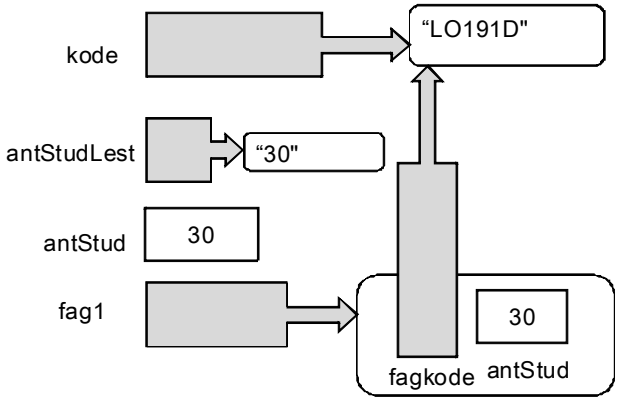
Oppgave 6

```
class Matvare {
  private final String navn;
```

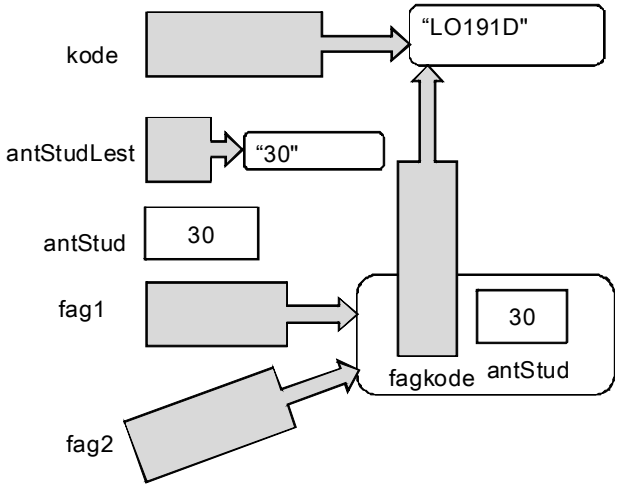
```
    private final double energi;  // pr gram
    private final double fett; // pr gram
    private final double karbohydr; // pr gram

    /**
     * Konstruktøren tar data for 100 gram av varen.
     */
    public Matvare(String navn, int energi, double fett, double karbohydr) {
      this.navn = navn;
      this.energi = energi * 0.01;
      this.fett = fett * 0.01;
      this.karbohydr = karbohydr * 0.01;
    }

    public String getNavn() {
      return navn;
    }

    public double finnFett(double gram) {
      return fett * gram;
    }

    public double finnKarbohydr(double gram) {
      return karbohydr * gram;
    }

    public double finnEnergiKJ(double gram) {
      return energi * gram;
    }

    public double finnEnergiKCAL(double gram) {
      return energi * gram * 4.18;
    }
  }
```

Kapittel 5.3

Oppgave 1



Oppgave 2



Oppgsave 3

Siste del av programmet ser nå slik ut:

```
Fag fag2 = fag1; // fra oppgave 2
fag1.setAntStud(fag1.getAntStud() + 5); // svar på oppgave 3
System.out.println("Antall studenter er " + fag2.getAntStud()); // kontroll
```

fag1 og fag2 peker til det samme objektet. Kan derfor bruke fag1 og fag2 om hverandre i de to siste linjene.

Oppgave 4

final medfører at vi ikke kan sette referansen fag3 til å peke til et annet objekt. Verdien til fag3 kan ikke forandres, den kan heller ikke settes null.

Datainnholdet i objektet kan endres.

Oppgave 5

For eksempel:

```
Fag fag4 = null;
fag4.setAntStud(100);
```

Kapittel 5.4

Oppgave 1

```
class Fag {
  private String fagkode = "Ukjent";  // må ta bort final pga set-metode
  private int antStud = 0;

  public Fag() { // ny konstruktør
  }

  public Fag(String fagkode) {  // ny konstruktør
    this.fagkode = fagkode;
  }

  public Fag(String fagkode, int antStud) {
    this.fagkode = fagkode;
    this.antStud = antStud;
  }

  public String getFagkode() {
    return fagkode;
  }

  public void setFagkode(String fagkode) {  // ny metode
    this.fagkode = fagkode;
  }

  public int getAntStud() {
    return antStud;
  }

  public void setAntStud(int antStud) {
    this.antStud = antStud;
  }
}
```

Kapittel 5.5

Metode:

```
public String toString() {
  return "Fagkode: " + fagkode + ", antall studenter: " + antStud + ".";
}
```

Utprøving:

```
public static void main(String[] args) {
  Fag etFag = new Fag("LO191D", 50);
  System.out.println(etFag);
}
```

Utskrift:

```
Fagkode: LO191D, antall studenter: 50.
```

Kapittel 6.1

Oppgave 1

Vi antar at rabatten regnes ut *før* momsen legges til, det vil si at det ikke er rabatt på momsen.

Vi antar videre at rabatten gis *fra og med* henholdsvis 3 og 5 kilo. Eventuelle feil som følge av beregninger med desimaltall tar vi ikke hensyn til, da det like gjerne kan gå den ene veien som den andre - og i det ene tilfellet kommer det kunden til gode, og i det andre tilfellet selgeren.

Nye konstanter:

```
public static final double RABATT1 = 10.0;
public static final double RABATTFAKTOR1 = (100.0 - RABATT1) / 100.0;
public static final double RABATTGRENSE1 = 3.0;
public static final double RABATT2 = 20.0;
public static final double RABATTFAKTOR2 = (100.0 - RABATT2) / 100.0;
public static final double RABATTGRENSE2 = 5.0;
```

Nye metoder:

```
public double finnPrisUtenMoms(double vekt) {
  double beregnetPris = vekt * pris;
  if (vekt >= RABATTGRENSE2) {
    beregnetPris *= RABATTFAKTOR2;
  } else if (vekt >= RABATTGRENSE1) {
    beregnetPris *= RABATTFAKTOR1;
  }
  return beregnetPris;
}

public double finnPrisMedMoms(double vekt) {
  return finnPrisUtenMoms(vekt) * MOMSFAKTOR;
}
```

Oppgave 2

```
public static void main(String[] args) {
  Vare vare = new Vare("Norgesost", 124);
  String kgPrisLest = showInputDialog("Kilopris, avslutt med Esc: ");
  while (kgPrisLest != null) {
    double kgPris = Double.parseDouble(kgPrisLest);
```

```
      String antKgLest = showInputDialog("Antall kilo: ");
      double antKg = Double.parseDouble(antKgLest);
      vare.setPris(kgPris);
      double prisUMoms = vare.finnPrisUtenMoms(antKg);
      double prisMMoms = vare.finnPrisMedMoms(antKg);
      java.util.Formatter f1 = new java.util.Formatter();
      f1.format("%.2f", prisUMoms);
      java.util.Formatter f2 = new java.util.Formatter();
      f2.format("%.2f", prisMMoms);
      java.util.Formatter f3 = new java.util.Formatter();
      f3.format("%.2f", Vare.MOMS);
      String melding = "Uten moms, kr " + f1.toString() + ", med moms kr "
                                    + f2.toString() + ", momsen er " + f3.toString() + " %.";
      showMessageDialog(null, melding);
      kgPrisLest = showInputDialog("Kilopris, avslutt med Esc: ");
    }
  }
```

Kapittel 6.2

Oppgavene 1–11:

1: Riktig.

2: Riktig.

3: Feil.

4: Riktig.

5: Feil, litt upresis formulering i oppgaveteksten. Det bør i stedet stå: Vi må bruke delete for å frigjøre lagerplass på heapen som ikke lenger er i bruk.

6: Ja. Også her burde oppgaveformuleringen vært mer presis: "Parameteroverføring endrer størrelsen på den delen av stakken som programmet bruker."

7: Feil.

8: Feil.

9: Riktig.

10: Feil.

11: Feil.

Kapittel 6.4

Oppgave 1

Hjelpemetoden ser slik ut:

```
  private String formaterDesimaltall(double tall) {
    java.util.Formatter formaterer =  new java.util.Formatter(); // formaterer utskriften
    formaterer.format("%.2f", tall);
    return formaterer.toString();
  }
```

Ny utgave av skrivUtNettolønn():

```
  public void skrivUtNettolønn(double netto) {
    showMessageDialog(null, formaterDesimaltall(netto));
```

```
}
```

Oppgave 2

Hjelpemetode:

```
/*
 * Leser et tall fra brukeren. Tallet må ligge i intervallet [nedre, ovre].
 * Hvis ikke, gis en feilmelding, og brukeren må skrive inn tallet på nytt.
 */
private double lesTall(String melding, double nedre, double ovre) {
  double tall = lesDesimaltall(melding
                    + "Et tall i intervallet [" + nedre + ", " + ovre + "]: ");
  while (tall < nedre || tall > ovre) {
    tall = lesDesimaltall("Tallet var utenfor oppgitt område: ["
                      + nedre + ", " + ovre + "]. Prøv på nytt : ");
  }
  return tall;
}
```

Ny utgave av registrerNySkatteprosent():

```
public void registrerNySkattepros() {
  double nyPros = lesTall("Ny skatteprosent. ", 0.0, Ansatt.MAKS_SK_PROS);
  enAnsatt.setSkatteprosent(nyPros);
}
```

Kapittel 6.5

I d

De to første punktene:

Vi legger inn en ny konstant:

```
public static final double MIN_PRIS = 0.50;
```

Konstruktøren som tar tre argumenter ser nå slik ut:

```
public Vare(String varenavn, int varenr, double pris) {
  if (varenr < 10000 || varenr > 99999) {
    throw new IllegalArgumentException(
    "Varenummeret skal være et femsifret tall. Her var det: " + varenr);
  }
  if (pris < MIN_PRIS) {
    throw new IllegalArgumentException(
    "Prisen må være minst kr " + MIN_PRIS + ". Her var det: " + pris);
  }
  this.varenavn = varenavn;
  this.varenr = varenr;
  this.pris = pris;
}
```

Metoden setPris() blir slik:

```
public void setPris(double pris) {
  if (pris < MIN_PRIS) {
    throw new IllegalArgumentException(
    "Prisen må være minst kr " + MIN_PRIS + ". Her var det: " + pris);
```

```
    }
    this.pris = pris;
  }
```

Det er videre et spørsmål om konstruktøren som tar to argumenter skal være gyldig, i og med at det setter prisen lik 0. Vi velger å beholde denne, men setter prisen lik MIN_PRIS:

```
  public Vare(String varenavn, int varenr) {
    if (varenr < 10000 || varenr > 99999) {
      throw new IllegalArgumentException(
      "Varenummeret skal være et femsifret tall. Her var det: " + varenr);
    }
    this.varenavn = varenavn;
    this.varenr = varenr;
    this.pris = MIN_PRIS;
  }
```

Det tredje punktet:

Antall kilo skal være mellom 0,1 og 10.

To nye konstanter:

```
  public static final double VEKTGRENSE1 = 0.1;
  public static final double VEKTGRENSE2 = 10.0;
```

Dette løser vi ved å la metodene som beregner prisen returnere en negativ verdi dersom argumentet er utenfor intervallet [0.1, 10.0]. Det er nok å legge inn testen i metoden finnPrisUtenMoms():

```
  public double finnPrisUtenMoms(double vekt) {
    if (vekt >= VEKTGRENSE1 && vekt <= VEKTGRENSE2) {
      double beregnetPris = vekt * pris;
      if (vekt >= RABATTGRENSE2) {
        beregnetPris *= RABATTFAKTOR2;
      } else if (vekt >= RABATTGRENSE1) {
        beregnetPris *= RABATTFAKTOR1;
      }
      return beregnetPris;
    } else {
      return -1.0;
    }
  }
```

Verdiene som returneres vil være forskjellige avhengig av hvilken av metodene klienten kaller, men det spiller ingen rolle. Det eneste han trenger å vite er at det er noe galt dersom verdien er negativ.

Kapittel 6.6

Datasett nr.	Vekt	Pris uten moms	Pris med moms
1	0.0	0.0	0.0
2	2.0	200.0	248.0
3	3.0	270.0	334.8
4	4.0	360.0	446.4
5	5.0	400.0	496.0
6	6.0	480.0	592.2

Kode:

```
public static void main(String[] args) {
  final double TOLERANSE = 0.001;
  System.out.println("Totalt antall tester: 6");
  Vare vare = new Vare("Ost", 124, 100);
  if (Math.abs(vare.finnPrisUtenMoms(0.0) - 0.0) < TOLERANSE &&
      Math.abs(vare.finnPrisMedMoms(0.0) - 0.0) < TOLERANSE) {
    System.out.println("Vare: Test 1 vellykket");
  }
  if (Math.abs(vare.finnPrisUtenMoms(2.0) - 200.0) < TOLERANSE &&
      Math.abs(vare.finnPrisMedMoms(2.0) - 248.0) < TOLERANSE) {
    System.out.println("Vare: Test 2 vellykket");
  }
  if (Math.abs(vare.finnPrisUtenMoms(3.0) - 270.0) < TOLERANSE &&
      Math.abs(vare.finnPrisMedMoms(3.0) - 334.8) < TOLERANSE) {
    System.out.println("Vare: Test 3 vellykket");
  }
  if (Math.abs(vare.finnPrisUtenMoms(4.0) - 360.0) < TOLERANSE &&
      Math.abs(vare.finnPrisMedMoms(4.0) - 446.4) < TOLERANSE) {
    System.out.println("Vare: Test 4 vellykket");
  }
  if (Math.abs(vare.finnPrisUtenMoms(5.0) - 400.0) < TOLERANSE &&
      Math.abs(vare.finnPrisMedMoms(5.0) - 496.0) < TOLERANSE) {
    System.out.println("Vare: Test 5 vellykket");
  }
  if (Math.abs(vare.finnPrisUtenMoms(6.0) - 480.0) < TOLERANSE &&
      Math.abs(vare.finnPrisMedMoms(6.0) - 595.2) < TOLERANSE) {
    System.out.println("Vare: Test 6 vellykket");
  }
```

Kapittel 6.7

Oppgave 1

Vi skal regne med gram og øre i stedet for kilo og kroner. Klienten skal fortsatt se kilo og kroner.

Klassen Vare ser nå slik ut:

```
class Vare {
  public static final double MOMS = 24.0;
  public static final double MOMSFAKTOR = 1.0 + MOMS / 100.0;
  public static final double RABATT1 = 10.0;
  public static final double RABATTFAKTOR1 = (100.0 - RABATT1) / 100.0;
  public static final double RABATTGRENSE1 = 3.0;
  public static final double RABATT2 = 20.0;
  public static final double RABATTFAKTOR2 = (100.0 - RABATT2) / 100.0;
  public static final double RABATTGRENSE2 = 5.0;

  /*
   * Faktorene multipliseres med 100 for å få heltall.
   * Grensene multipliseres med 1000 for å få gram.
   */
  private static final int I_MOMSFAKTOR = (int) (MOMSFAKTOR * 100.0);
  private static final int I_RABATTFAKTOR1 = (int) (RABATTFAKTOR1 * 100.0);
  private static final int I_RABATTGRENSE1 = (int) (RABATTGRENSE1 * 1000.0);
  private static final int I_RABATTFAKTOR2 = (int) (RABATTFAKTOR2 * 100.0);
  private static final int I_RABATTGRENSE2 = (int) (RABATTGRENSE2 * 1000.0);

  private final String varenavn;
  private final int varenr;
  private int pris; // pris i øre pr. gram, alltid uten moms

  public Vare(String varenavn, int varenr, double pris) {
    this.varenavn = varenavn;
    this.varenr = varenr;
    this.pris = regnOmTilØrePrGram(pris);
  }

  public Vare(String varenavn, int varenr) {
    this.varenavn = varenavn;
    this.varenr = varenr;
    this.pris = 0;
  }

  public String getVarenavn() {
    return varenavn;
  }

  public int getVarenr() {
    return varenr;
  }
```

```
public double getPris() {
  return regnOmTilKrPrKg(pris);
}

public void setPris(double pris) {
  this.pris = regnOmTilØrePrGram(pris);
}

/**
 *
 * Finner pris uten moms. Eventuell rabatt beregnes.
 */
public double finnPrisUtenMoms(double vekt) {
  int vektGram = regnOmTilGram(vekt);
  int beregnetPris = vektGram * pris;
  /*
   * For å beregne rabattert pris, må vi dividere med 100.
   * Her kan vi miste nøyaktighet på grunn av at dette er heltallsdivisjon.
   */
  if (vektGram >= I_RABATTGRENSE2) {
    beregnetPris = beregnetPris * I_RABATTFAKTOR2 / 100;
  } else if (vektGram >= I_RABATTGRENSE1) {
    beregnetPris = beregnetPris * I_RABATTFAKTOR1 / 100;
  }
  return regnOmTilKr(beregnetPris);
}

/**
 *
 * Finner pris med moms.
 */
public double finnPrisMedMoms(double vekt) {
  // Litt søkt å konvertere til og fra heltall her på grunn av at det
  // samme skjer i finnPrisUtenMoms(),
  // men oppgaven sier "kun" heltallsberegninger, så da så.
  int prisUtenMomsØre = regnOmTilØre(finnPrisUtenMoms(vekt));
  int prisMedMomsØre = prisUtenMomsØre * I_MOMSFAKTOR / 100;
  return regnOmTilKr(prisMedMomsØre);
}

public String toString() {
  java.util.Formatter f = new java.util.Formatter();
  f.format("%.2f", regnOmTilKrPrKg(pris));
  return varenr + ": " + varenavn + ", pris pr. kg kr " + f.toString() + " u.moms.";
}

/*
 * Hjelpemetoder.
 */
private int regnOmTilØre(double kr) {
  return (int) (kr * 100);
```

```
  }

  private double regnOmTilKr(long øre) {
    return øre * 0.01;
  }

  private int regnOmTilØrePrGram(double kr) {
    double øre = regnOmTilØre(kr);  // øre pr. kg
    return (int) (øre * 0.001);  // øre pr. gram
  }

  private double regnOmTilKrPrKg(long øre) {
    double kr = regnOmTilKr(øre); // kr pr gram
    return kr * 1000;  // kr pr. kilo
  }

  private int regnOmTilGram(double kg) {
    return (int) (kg * 1000);
  }

  private double regnOmTilKg(int gram) {
    return gram * 0.001;
  }
```

Kapittel 7.1

Oppgave 1

```
import static javax.swing.JOptionPane.*;
class Oppg7_1_1 {
  public static void main(String[] args) {

    /* Oppgave a */
    int[] antDager = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};

    /* Oppgave b */
    String svar = showInputDialog("Er det skuddår? Svar j(a) eller n(ei): ");
    char bokstav = svar.charAt(0); // i tilfelle brukeren skriver ordene helt ut
    if (bokstav == 'j' || bokstav == 'J') {
      antDager[1] = 29;
    }

    /* Testutskrift */
    for (int i = 0; i < antDager.length; i++) {
      System.out.println(antDager[i]);
    }
  }
}
```

Oppgave 2

```
char[] navn = {'A', 'N', 'N', 'E'};
for (int i = navn.length - 1; i >= 0; i--) {
  System.out.print(navn[i]); // ikke linjeskift
```

```
}
System.out.println(); // et linjeskift til slutt
```

Oppgave 3

Tabellen har følgende innhold: 3, 8, 8, 3, 7, 14, 3, 17, 8, 9

Kapittel 7.2

Oppgave 1

```
int tabLengde = tab1.length; // lik for begge tabellene
for (int i = 0; i < tabLengde; i++) {
  tab2[tabLengde - 1 - i] = tab1[i];
}
```

Oppgave 2

Utskriften blir som følger:

```
novemberr
maiai
```

Kapittel 7.3

Oppgave 1

```
public int finnAntDgMaks() {
  int maks = finnMaksimum();
  int antMaks = 0;
  for (int i = 0; i < nedbør.length; i++) {
    if (nedbør[i] == maks) {
      antMaks++;
    }
  }
  return antMaks;
}
```

Utprøving: Legg følgende inn i klientprogrammet:

```
int antDgMaks = januar.finnAntDgMaks();
System.out.println("Antall dager med maks. nedbør: " + antDgMaks);
```

Oppgave 2

```
public int antDgMindreEnn(int grense) {
  int antall = 0;
  for (int i = 0; i < nedbør.length; i++) {
    if (nedbør[i] < grense) {
      antall++;
    }
  }
  return antall;
}
```

Utprøving:

```
int antDgMindreEnnGrense = januar.antDgMindreEnn(4);
System.out.println("Antall dager med nedbør mindre enn 4: "
                   + antDgMindreEnnGrense);
```

Oppgave 3

```
public double finnStdAvvik() {
  /*
   * Metoden finnGjSnitt() runder av gjennomsnittet til nærmeste heltall.
   * Resultatet nedenfor hadde blitt mer nøyaktig om vi hadde beholdt
   * gjennomsittsverdien som desimaltall.
   */
  if (nedbør.length > 1) {
    int gjSnitt = finnGjSnitt();
    long sumKvadrat = 0;
    for (int i = 0; i < nedbør.length; i++) {
      sumKvadrat += (gjSnitt - nedbør[i]) * (gjSnitt - nedbør[i]);
    }
    System.out.println(sumKvadrat);
    double radikand = (double) sumKvadrat / (double) (nedbør.length - 1);
    return Math.sqrt(radikand);   // Retur. Std.avvik beregnet etter formel
  } else {
    return 0.0;              // Retur. For lite data til å beregne std.avvik
  }
}
```

Utprøving:

```
double stdAvvik = januar.finnStdAvvik();
System.out.println("Standard avvik er: " + stdAvvik);
```

Oppgave 4

```
public Maned(String mndNavn, int[] nedbør) {
  /*
   * Sjekker først at ingen av nedbørverdiene er negative.
   */
  for (int i = 0; i < nedbør.length; i++) {
    if (nedbør[i] < 0) {
      throw new IllegalArgumentException(
      "\nNedbørverdi med indeks " + i + " var negativ: " + nedbør[i]
          + ". \nDet kan finnes flere negative verdier.");
    }
  }
  /*
   * Hvis vi kommer hit, er dataene ok, og de lagres i objektet.
   */
  this.mndNavn = mndNavn;
  int antDager = nedbør.length;
  this.nedbør = new int[antDager];
  for (int i = 0; i < antDager; i++) {
    this.nedbør[i] = nedbør[i];
  }
}
```

Utprøving ved normal kjøring (ingen feilmelding) og ved at verdier i tabellen nedbør settes negativ (feilmelding med programstopp):

```
int[] nedbør = {1, -4, 0, 4, 3};
```

Kapittel 7.4

Oppgave 1

Utskrift "dyp" utgave:

```
Utskrift 1: maksimum: 4, ant. tørre dager: 2
Utskrift 2: maksimum: 4, ant. tørre dager: 2
Utskrift 3: maksimum: 4, ant. tørre dager: 2
```

Utskrift "grunn" utgave:

```
Utskrift 1: maksimum: 4, ant. tørre dager: 2
Utskrift 2: maksimum: 10, ant. tørre dager: 2
Utskrift 3: maksimum: 10, ant. tørre dager: 3
```

Kapittel 7.5

Oppgave 1

Kompileringsfeil: "missing return statement"

Kompilatoren krever at alle mulige "veier" gjennom en metode har en return-setning (hvis metoden ikke er void). I dette tilfellet mangler return-setning i det tilfellet at man ikke kommer inn i for-løkken. (Kompilatoren kan ikke vite at nedbør.length alltid er >= 0.)

Logisk feil: Metoden fungerer kun dersom verdien vi søker, ligger aller først i tabellen. Nummerér linjene 1–9. I første løkkegjennomløp er i = 0. Hvis nedbør[0] ikke er lik verdi, vil else-delen av if-setningen utføres, og i linje 6 hopper vi ut av metoden med -1 som verdi.

Løsningen må være som metoden finnDag() gitt først i oppgaveteksten. Eventuelt skriver vi om med en while-setning som forklart i teksten.

Oppgave 2

Med konstruksjonen

```
while (dagNr < nedbør.length && !funnet) {
```

ser metoden finnDag() slik ut:

```
public int finnDag(int verdi) {
  boolean funnet = false;
  int dagNr = 0;
  while (dagNr < nedbør.length && !funnet) {
    if (nedbør[dagNr] == verdi) {
      funnet = true;
    } else {
      dagNr++; // bare økning dersom ikke funnet
    }
  }
  if (funnet) {
    return dagNr;
  } else {
   return -1;
```

```
  }
}
```

Hvis det eksisterer flere forekomster av verdi i tabellen, finner også denne varianten (som i oppgave 1) av finnDag() indeksen til den *første* av disse forekomstene.

Vi endrer betingelsen til følgende:

```
while (dagNr < nedbør.length) {
```

Hvis verdien ikke finnes i tabellen, fungerer koden fortsatt som den skal.

Hvis verdien finnes i tabellen, får vi en evig løkke. Når programkontrollen støter på verdien, blir ikke dagNr oppdatert – og dermed blir aldri løkkebetingelsen usann.

Vi skal skrive om finnDag() slik at vi kan bruke betingelsen foran. Da forsvinner også den logiske variabelen funnet:

```
public int finnDag(int verdi) {
  int dagNr = 0;
  int indeksVerdi = -1;  // lagrer indeksen her
  while (dagNr < nedbør.length) {
    if (nedbør[dagNr] == verdi) {
      indeksVerdi = dagNr;
    }
    dagNr++;
  }
  return indeksVerdi;
}
```

Merk at metoden her returnerer indeksen til *siste* forekomst av verdien.

Kapittel 8.3

Oppgave 1

```
public static void main(String[] args) {
  String[] navn = {"Toril", "Ole", "Petter", "Kari"};

  String alleNavn = "";
  for (int i = 0; i < navn.length - 1; i++) {
    alleNavn += navn[i] + ", ";
  }
  if (navn.length >= 1) {
    alleNavn += navn[navn.length - 1]; // ikke komma etter det siste
  }
  showMessageDialog(null, alleNavn);
}
```

Oppgave 2

```
String[] navn = new String[4];
for (int i = 0; i < navn.length; i++) {
  String lestNavn = showInputDialog("Skriv navn nr. " + (i + 1) + ": ");
  navn[i] = lestNavn.trim();
}
```

Kapittel 8.4

Oppgave 1

```
import java.util.Random;
class Oppg8_4_1 {
  public static void main(String[] args) {
    Random randomGen = new Random();

    /* Oppgave a) */
    long tall1 = randomGen.nextLong();
    long tall2 = randomGen.nextLong();
    long tall3 = randomGen.nextLong();
    System.out.println("Tre tilfeldige tall, long: " +
            tall1 + ", " + tall2 + ", " + tall3);

    /* Oppgave b) */
    int talla = randomGen.nextInt(1001);
    int tallb = randomGen.nextInt(1001);
    int tallc = randomGen.nextInt(1001);
    System.out.println("Tre tilfeldig heltall [0..1000]: " +
            talla + ", " + tallb + ", " + tallc);

    /* Oppgave c) */
    double desTall1 = randomGen.nextDouble();
    double desTall2 = randomGen.nextDouble();
    System.out.println("To tilfeldige desimaltall [0.0..1.0>: "
                        + desTall1 + ", " + desTall2);

    /* Oppgave d) */
    int tallA = (randomGen.nextInt(1001) - 500);
    int tallB = (randomGen.nextInt(1001) - 500);
    System.out.println("To tilfeldige heltall [-500..500]: " + tallA + ", " + tallB);

    /* Oppgave e) */
    double tallC = 100.0 * randomGen.nextDouble();
    double tallD = 100.0 * randomGen.nextDouble();
    System.out.println("To tilfeldige desimaltall [0.0..100.0>: " + tallC + ", " + tallD);
  }
}
```

Kapittel 8.5

Oppgave 1

```
class Oppg8_6_1 {
  public static void main(String[] args) {
    String test1 = "endelser";
    String test2 = "endelsene";
    if (test1.endsWith("en")) {
      System.out.println(test1 + " ender på \"en\".");
    } else {
      System.out.println(test1 + " ender ikke på \"en\".");
```

```
    }
    if (test2.endsWith("ene")) {
      System.out.println(test2 + " ender på \"ene\".");
    } else {
      System.out.println(test2 + " ender ikke på \"ene\".");
    }

    // plass til begge ordene med mellomrom:
    char[] bokstaver = new char[test1.length() + test2.length() + 1];
    test1.getChars(0, test1.length(), bokstaver, 0);
    bokstaver[test1.length()] = ' '; // legger inn mellomrommet
    test2.getChars(0, test2.length(), bokstaver, test1.length() + 1);
    for (int i = 0; i < bokstaver.length; i++) {
      System.out.print(bokstaver[i]);
    }
    System.out.println();

    if (test1.isEmpty()) {
      System.out.println("test1 er tom");
    } else {
      System.out.println("test1 er ikke tom");
    }
    String test3 = "";
    if (test3.isEmpty()) {
      System.out.println("test3 er tom");
    } else {
      System.out.println("test3 er ikke tom");
    }
  }
}
```

Utskriften blir som følger:

```
endelser ender ikke på "en".
endelsene ender på "ene".
endelser endelsene
test1 er ikke tom
test3 er tom
```

Oppgave 2

Kompilerer følgende program:

```
class Oppg8_6_2 {
  public static void main(String[] args) {
    java.util.Date dato = new java.util.Date(2002, 8, 1);  // deprecated konstruktør
    System.out.println(dato);
  }
}
```

Vi får følgende melding:

```
Note: J:\TEMP\Oppg8_6_2.java uses or overrides a
deprecated API.
Note: Recompile with -Xlint:deprecation for details.
```

Dersom vi kompilerer på nytt med opsjonen -Xlint:deprecation får vi nærmere beskjed:

```
J:\TEMP\Oppg8_6_2.java:4: warning: Date(int,int,int)
in java.util.Date has been deprecated
java.util.Date dato = new java.util.Date(2002, 8, 1);
```

NB! Programmet kan kjøres selv om det bruker en deprecated konstruktør. Utskriften blir som følger:

```
Mon Sep 01 00:00:00 CEST 3902
```

Kapittel 8.7

Oppgave 1

```
Integer etHeltall = new Integer(15); // oppgave a)
Character forbokstav = new Character('S'); // oppgave b)
int tall = etHeltall.intValue(); // oppgave c)
String tekst1 = forbokstav.toString(); // oppgave d)
String tekst2 = etHeltall.toString(); // oppgave e)
```

Oppgave 2

Siste del av oppgaveteksten krever at du behersker unntakshåndtering. Det er tema i kapittel 15.

```
try {
  String tekst = "123456.67";
  int tall = Integer.parseInt(tekst);
  System.out.println("Tallet er " + tall);
} catch (NumberFormatException e) {
  System.out.println("Kan ikke omforme denne teksten til tall.");
}
```

Kapittel 8.9

Oppgave 1

a) Fra linje 21 og utover:

```
if (antOrd > 0) {
  double gjsnitt = (double) antBokst / (double) antOrd;  // oppgave a)
  int lengde = tekst.length();
  int antSkilletegn = lengde - antBokst;
  showMessageDialog(null, "Antall ord er " + antOrd +
            ".\nGjennomsnittling ordlengde er " + gjsnitt + "\nTekstlengde: " +
      lengde + "\nAntall skilletegn: " + antSkilletegn +
      "\nAndelen skilletegn er " + (double) antSkilletegn / (double) lengde);
  } else {
    showMessageDialog(null, "Ingen ord skrevet inn.");
  }
```

b) Linje 13:

```
String skilletegn = "0123456789" + showInputDialog("Skilletegn: ");
```

Oppgave 2

```
import static javax.swing.JOptionPane.*;
class Oppg8_9_2 {
  public static void main(String[] args) {
    String tekst = showInputDialog("Teksten: ");
    String skilletegn = showInputDialog("Skilletegn: ");
    String[] ord = tekst.split("[" + skilletegn + "]");
    int antOrd = 0;
    int antBokst = 0;
    for (int i = 0; i < ord.length; i++) {
      int lengde = ord[i].length();
      if (lengde > 0) {
        antBokst += lengde;
        antOrd++;
      }
    }
    if (antOrd > 0) {
      double gjsnitt = (double) antBokst / (double) antOrd;
      showMessageDialog(null, "Antall ord er " + antOrd
                        + ".\nGjennomsnittling ordlengde er " + gjsnitt);
    } else {
      showMessageDialog(null, "Ingen ord skrevet inn.");
    }
  }
}
```

Kapittel 8.12

Oppgave 1

```
class Oppg8_12_1 {
  public static void main(String[] args) {
    java.util.Locale tysk = new java.util.Locale("de", "DE");
    java.util.Locale.setDefault(tysk);
    double beløp = 3.467;

    /* Lager streng der tallet vises med én desimal. */
    java.util.Formatter f = new java.util.Formatter();
    f.format("%.1f", beløp);

    /* Myntformatering krever bruk av klassen NumberFormat, se side 679. */
    java.text.NumberFormat myntFormat
                                = java.text.NumberFormat.getCurrencyInstance();
    String utskrift = "Med en desimal: " + f.toString()
                    + ", i euro " + myntFormat.format(beløp);

    /* Utskrift i meldingsboks. */
    javax.swing.JOptionPane.showMessageDialog(null, utskrift);

    /* Utskrift i komm. vinduet, euro-tegnet blir ikke riktig i Windows-komm. vindu. */
    System.out.println(utskrift);
```

```
  }
}
```

Oppgave 2

```
for (int i = 1; i <= 10; i++) {
  for (int j = 1; j <= 10; j++) {
    System.out.printf("%2dx%2d=%2d ", i, j, i * j);
  }
  System.out.println();
}
```

Oppgave 3

```
Date nå = new Date();
System.out.printf("I dag er det %1$tA. Akkurat nå er klokka %1$tH:%1$tM, " +
                  "datoen er %1$td. %1$tB %1$tY\n", nå);
```

Kapittel 9.1

Oppgave 1

Kommentar: Det er et brudd med Java-kodestil i boka, linje 14, programliste 9.1, side 288. Linje 14 bør skiftes ut med:

```
if (tabell[i] < tabell[hittilMinst]) {
  hittilMinst = i;
}
```

Tilbake til oppgaven: Vi må snu ulikhetstegnet i sammenlikningen, og vi bør helst også skifte navn på variabelen HittilMinst til HittilStørst:

```
public static void sorterHeltallstabell2(int[] tabell) {
  for (int start = 0; start < tabell.length; start++) {
    int hittilStørst = start;
    for (int i = start + 1; i < tabell.length; i++) {
      if (tabell[i] > tabell[hittilStørst]) {
        hittilStørst = i;
      }
    }
    int hjelp = tabell[hittilStørst];
    tabell[hittilStørst] = tabell[start];
    tabell[start] = hjelp;
  }
}
```

Oppgave 2

Klasse med metode sorterOgFjernDubletter() og meget enkel testklient:

```
class Oppg9_1_2 {
  public static int[] sorterOgFjernDubletter(int[] tab) {
    Sortering.sorterHeltallstabell(tab);  // sorterer først tabellen, programliste 9.1
    if (tab.length > 0) {
      int[] nyTab = new int[tab.length];
      nyTab[0] = tab[0];
```

```
    int antall = 1; // antall forskjellige tall
    int forrigeTall = tab[0];
    int indeks = 1; // indeks i sortertTab

    /*
     * Løper gjennom hele tabellen.
     */
    while (indeks < tab.length) {
      /* hopper over alle dubletter */
      while (indeks < tab.length && tab[indeks] == forrigeTall) {
        indeks++;
      }

      /*
       * Hvis vi ikke har nådd tabell-slutt,
       * har vi funnet et element med en annen verdi.
       */
      if (indeks < tab.length) {
        forrigeTall = tab[indeks];
        nyTab[antall] = tab[indeks];
        antall++;
        indeks++;
      }
    }

    /*
     * Lager en tabell med akkurat riktig størrelse.
     */
    int[] nyTab2 = new int[antall];
    for (int i = 0; i < antall; i++) {
      nyTab2[i] = nyTab[i];
    }
    return nyTab2;
  }
  return null; // parameteren refererer til en tom tabell
}

public static void main(String[] args) {
  int[] test = {3, 4, -5, 13, 10, 4, 11, 0, 8, -2, 11, 22, 15, 11, 9, 17, 4};
  int[] tab = sorterOgFjernDubletter(test);
  System.out.println("Sortert tabell, dubletter fjernet: ");
  for (int i = 0; i < tab.length; i++) System.out.print(tab[i] + " ");
  System.out.println();
}
}

/* Kjøring av programmet:
Sortert tabell, dubletter fjernet:
-5 -2 0 3 4 8 9 10 11 13 15 17 22
*/
```

Oppgave 3

```
public static int søk(int[] tab, int verdi) {
  int indeks = 0;
  while (indeks < tab.length && tab[indeks] <= verdi) {
    if (tab[indeks] == verdi) {
      return indeks;
    }
    indeks++;
  }
  return -1;
}
```

Oppgave 4

```
public static int[] adderTabeller(int[] tab1, int[] tab2) {
  if (tab1.length == tab2.length) {
    int[] sum = new int[tab1.length];
    for (int i = 0; i < tab1.length; i++) {
      sum[i] = tab1[i] + tab2[i];
    }
    return sum;  // retur
  }
  return null; // retur, tabellene ikke like lange
}
```

Kapittel 9.2

Oppgave 1

Metodehodet kan se slik ut:

```
public static int binærsøkHeltallstabell2(int[] tabell, int verdi, int antElementer) {
```

Linje 16 skiftes ut med følgende:

```
int slutt = antElementer - 1;
```

Oppgave 2

```
public static void main(String[] args) {
  int[] enTabell = {-15, -5, 5, 7, 17, 0, 0, 0};
  int antElementer = 5;
  String søkeverdiLest = showInputDialog("Oppgi søkeverdi, avslutt med Esc: ");
  while (søkeverdiLest != null) {
    int søkeverdi = Integer.parseInt(søkeverdiLest);
    int returverdi = binærsøkHeltallstabell2(enTabell, søkeverdi, antElementer);
    if (returverdi >= 0) {
      System.out.println("Verdien er funnet på indeks " + returverdi + ".");

    } else {
      if (antElementer < enTabell.length) {  // plass i tabellen
        /*
         * Skal legge inn søkeverdien på riktig plass i tabellen.
         * Flytter resten av elementene en plass utover i tabellen.
         * Legger deretter inn det nye elementet og øker antElementer.
         * Endelig skrives tabellen ut for kontroll.
         */
        int indeks = -returverdi - 1;  // skal legges inn på denne indeks
```

```
        for (int i = antElementer; i > indeks; i--) {
          enTabell[i] = enTabell[i - 1];
        }
        enTabell[indeks] = søkeverdi;
        antElementer++;
        System.out.print("Oppdatert tabell: ");
        for (int i = 0; i < antElementer; i++) {
          System.out.print(enTabell[i] + " ");
        }
        System.out.println();

      } else {
        System.out.println(
              "Verdien ikke funnet, og heller ikke plass til å legge den inn.");
      }
    }
    søkeverdiLest = showInputDialog("Oppgi søkeverdi, avslutt med Esc: ");
  }
}
```

Kapittel 9.4

```
/**
 * Fjerner alle forekomster av et tall fra tabellen.
 * Et tall fjernes ved at siste element i tabellen legges på
 * tallets plass - slik at tabellen i neste omgang kan krympes
 * ved å fjerne elementer bakerst.
 */
public void fjernTall(int tall) {
  int antTallFjernet = 0;
  /* Begynner bakfra fordi antallTall endres */
  for (int i = antallTall - 1; i >= 0; i--) {
    if (tabell[i] == tall) {  // tall skal fjernes
      if (antallTall > 1) {
        tabell[i] = tabell[antallTall - 1];
      }
      antallTall--;
      antTallFjernet++;
    }
  }
  if (antTallFjernet > 2) {
    krympTabell();
  }
}

/*
 * Hjelpemetode for å "krympe" tabellen
 * Forutsetter at tabellen kan krympes bakfra.
 */
private void krympTabell() {
  if (tabell.length > 2) {
    int[] nyTab = new int[tabell.length - 2];  // ny og mindre tabell
```

```
      for (int i = 0; i < nyTab.length; i++) {  // kopierer data over til ny tabell
        nyTab[i] = tabell[i];
      }
      tabell = nyTab;  // setter objektvariabelen lik den nye tabellen
    }
  }
```

Kapittel 9.5

Oppgave 1

Ukenummerne ligger én forskjøvet i forhold til indeksene, eksempel: Uke 10 har indeks 9. Tilsvarende for dagene: Dagen med indeks 0 er mandag.

```
class Oppg9_5_1 {
  public static void main(String[] args) {
    Salgstall året2009 = new Salgstall("sko", 52, 5);  // oppgave a)
    året2009.settSalg(9, 0, 10000);  // oppgave b)
    året2009.settSalg(7, 3, 12100);  // oppgave c)
    System.out.println("Total salg uke 5: " + året2009.finnSalgForHelUke(4)); // d)
    System.out.println("Salg uke 6, mandag: " + året2009.finnSalg(5, 0));  // e)
    System.out.println("Totalt salg hele året: " + året2009.finnTotalsalg());  // f)
    int dag = året2009.finnMestLønnsommeUkedag();  // oppgave g)
    System.out.println("Mest lønnsomme ukedag: Dag nr. " + (dag + 1)
                       + " med salg på: " + året2009.finnSalgForUkedag(dag));
  }
}

/* Kjøring av programmet:
Total salg uke 5: 0
Salg uke 6, mandag: 0
Totalt salg hele året: 22100
Mest lønnsomme ukedag: Dag nr. 4 med salg på: 12100
*/
```

Oppgave 2a

```
public double finnSnittPrUke() {
  int sum = 0;
  for (int i = 0; i < finnAntUker(); i++) {
    sum += finnSalgForHelUke(i);
  }
  if (finnAntUker() > 0) {
    return (double) sum / (double) finnAntUker();
  }
  return 0;
}
```

Oppgave 2b

```
public int[] finnLønnsommeUker() {
  if (finnAntUker() > 0) {

    /* Finner først maksimalverdien. */
    int maks = finnSalgForHelUke(0);
    for (int i = 1; i < finnAntUker(); i++) {
```

```
      if (finnSalgForHelUke(i) > maks) {
        maks = finnSalgForHelUke(i);
      }
    }

    /* Finner deretter alle ukene med denne verdien. */
    int[] ukenr = new int[finnAntUker()];  // tabell med plass til ukenr
    int antall = 0; // antall uker med verdi lik maks
    for (int i = 0; i < finnAntUker(); i++) {
      if (finnSalgForHelUke(i) == maks) {
        ukenr[antall] = i;
        antall++;
      }
    }

    /* Lager tabell av riktig størrelse, og kopierer over. */
    int[] ukenrMedMaksSalg = new int[antall];
    for (int i = 0; i < antall; i++) {
      ukenrMedMaksSalg[i] = ukenr[i];
    }
    return ukenrMedMaksSalg;
  }
  return null;
}
```

Oppgave 3a

Tabellen resultat:

```
int[][] resultat = new int[antlag][ANT_MULIGE_RES];
```

Antall linjer tilsvarer antall lag. Antall kolonner er lik ANT_MULIGE_RES.

Kolonne 0: antall kamper vunnet.

Kolonne 1. antall kamper uavgjort

Kolonne 2: antall kamper tapt

Oppgave 3b

```
class Serie {
  private final int ANT_MULIGE_RES = 3;
  private final int ANT_POENG_VUNNET = 3;
  private final int ANT_POENG_UAVGJORT = 1;
  private final int ANT_POENG_TAP = 0;

  /* Tabellen resultat:
   * Antall linjer tilsvarer antall lag.
   * Antall kolonner er lik antMuligeRes.
   * Kolonne 0: antall kamper vunnet.
   * Kolonne 1. antall kamper uavgjort
   * Kolonne 2: antall kamper tapt
   */

  private int[][] resultat;
```

```
  public Serie(int antlag) {
    resultat = new int[antlag][ANT_MULIGE_RES];
    resultat[3][0] = 1;   //for testing
    resultat[3][1] = 2;
    resultat[3][2] = 3;
    resultat[6][0] = 1;
    resultat[6][1] = 2;
    resultat[6][2] = 3;
  }

  public int[] finnAntPoeng() {
    int antlag = resultat.length;
    int[] poengtabell = new int[antlag];
    for (int i = 0; i < poengtabell.length; i++) {
      poengtabell[i] = resultat[i][0] * ANT_POENG_VUNNET
            + resultat[i][1] * ANT_POENG_UAVGJORT
            + resultat[i][2] * ANT_POENG_TAP;
    }
    return poengtabell;
  }
}

class Oppg9_5_3 {
  public static void main(String[] args) {
    Serie fotball = new Serie(10);
    int[] poeng = fotball.finnAntPoeng();
    for (int i = 0; i < poeng.length; i++) {
      System.out.println("Lag nr. " + i + ": " + poeng[i] + " poeng");
    }
  }
}
```

Kapittel 9.6

Oppgave 1

Vi har lagt inn endringene i begynnelsen av klassen SalgBGS:

```
class SalgBGS {

  /* Konstanter til bruk i menyen */
  private final String[] ALTERNATIVER = {"Registrer nytt salg",
  "Finn salg for dag", "Finn salg for uke", "Finn totalsalg", "Finn salg gitt dag",
  "Finn mest løsnnsomme dag", "Avslutt"};
  private final int REG_SALG = 0;   // tallverdiene svarer til indeksen i
  private final int FINN_SALG = 1;  // tabellen over
  private final int FINN_SALG_UKE = 2;
  private final int FINN_TOTALSALG = 3;
  private final int FINN_SALG_UKEDAG = 4;
  private final int FINN_LØNNSOMME_UKEDAG = 5;
  private final int AVSLUTT = 6;

  /* Objektet som vi skal kommunisere med */
```

```
private Salgstall salg;
public SalgBGS(Salgstall salg) {
  this.salg = salg;
}


/**
 * Viser menyen som nedtrekksliste.
 * Metoden returnerer brukerens valg som en indeks i tabellen ALTERNATIVER.
 * Hvis brukeren vil avslutte, returneres en negativ verdi.
 */
public int lesValg() {
  Object obj = showInputDialog(null, "Gjør et valg", "Salgstall",
              DEFAULT_OPTION, null, ALTERNATIVER, ALTERNATIVER[0]);
  int valg = finnValg(obj);
  if (valg == AVSLUTT) {
    valg = -1;
  }
  return valg;
}

/* Hjelpemetode som søker i tabellen ALTERNATIVER etter en valgt verdi. */
private int finnValg(Object valg) {
  for (int i = 0; i < ALTERNATIVER.length; i++) {
    if (ALTERNATIVER[i].equals(valg)) {
      return i;
    }
  }
  return AVSLUTT; // skal ikke komme hit
}

/**
 * Hovedflyt for utførelse av valgt oppgave.
 */
public void utførValgtOppgave(int valg) {
    switch (valg) {
    case REG_SALG:
      regSalg();
      break;
    case FINN_SALG:
      finnSalg();
      break;
    case FINN_SALG_UKE:
      finnSalgUke();
      break;
    case FINN_TOTALSALG:
      finnTotalsalg();
      break;
    case FINN_SALG_UKEDAG:
      finnSalgUkedag();
      break;
```

```
      case FINN_LØNNSOMME_UKEDAG:
        finnMestLønnsommeUkedag();
        break;
      default:
        break;
    }
  }

 /**
  *
  * Finner totalsalget på en bestemt ukedag.
  *
  * Innlesing fra brukeren: Dagnr.
  * Utskrift: Salgstall, eventuelt feilmelding.
  */
 public void finnSalgUkedag() {
   int maksDag = salg.finnAntDgPrUke();
   int dag = lesHeltall("Oppgi dagnr (min 1, maks " + maksDag + ")", 1, maksDag);
   int salget = salg.finnSalgForUkedag(dag - 1);
   showMessageDialog(null, "Salget på ukedag nr " + dag + " er kr " + salget + ".");
 }

 /**
  *
  * Finner hvilken ukedag som er den mest lønnsomme.
  *
  * Innlesing fra brukeren: Ingenting.
  * Utskrift: Ukedagnr.
  */
 public void finnMestLønnsommeUkedag() {
   int dag = salg.finnMestLønnsommeUkedag();
   showMessageDialog(null, "Mest lønnsomme ukedag er dag nr " + (dag + 1));
 }
```

Kapittel 9.7

Oppgave 1

```
class Oppg9_7_1 {
  public static void main(String[] args) {
    int[] mndlengde = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
    final int ANTMND = mndlengde.length;
    int[][] nedbør = new int[ANTMND][];
    for (int i = 0; i < ANTMND; i++) {
      nedbør[i] = new int[mndlengde[i]];
    }

    nedbør[1][2] = 5; // 3.februar
    nedbør[1][6] = 12; // 7.februar
    nedbør[1][17] = 10; // 18.februar
    nedbør[5][6] = 15; // 7.juni
    nedbør[5][9] = 20; // 10.juni
```

```
    System.out.print("Nedbør februar: ");
    int sum = 0;
    for (int i = 0; i < mndlengde[1]; i++) {
      sum += nedbør[1][i];
    }
    System.out.println(sum + " mm.");

    System.out.print("Nedbør juni: ");
    sum = 0;
    for (int i = 0; i < mndlengde[5]; i++) {
      sum += nedbør[5][i];
    }
    System.out.println(sum + " mm.");
  }
}
```

Kapittel 10.1

Oppgave 1

```
import java.awt.*;  // klassene Container, Color og Graphics
import javax.swing.*; // klassene JFrame og JPanel

class Vindu extends JFrame {
  public Vindu(String tittel) {
    setTitle(tittel);
    setSize(300, 200); // bredde, høyde
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    Tegning tegningen = new Tegning();
    add(tegningen);
  }
}

class Tegning extends JPanel {
  public void paintComponent(Graphics tegneflate) {
    super.paintComponent(tegneflate);  // Husk denne!
    setBackground(Color.WHITE);
    tegneflate.setColor(Color.RED);
    tegneflate.drawRect(40, 30, 150, 80); // x, y, bredde, høyde
    tegneflate.drawString("Her er løsningen", 50, 50);

  }
}

class Oppg10_1_1 {
  public static void main(String[] args) {
    Vindu etVindu = new Vindu("Oppgave 10-1-1");
    etVindu.setVisible(true);
  }
}
```

Kapittel 10.2

Oppgave 1a

```
class Vindu extends JFrame {
  public Vindu(String tittel) {
    setTitle(tittel);
    setSize(200, 200); // bredde, høyde - litt høyere enn i originalen
    setDefaultCloseOperation(EXIT_ON_CLOSE);
    Tegning tegningen = new Tegning();
    add(tegningen);
  }
}

class Tegning extends JPanel {
  private static final Font skrift = new Font("Dialog", Font.BOLD, 16);
  public void paintComponent(Graphics tegneflate) {
    super.paintComponent(tegneflate);  // husk denne!
    setBackground(Color.ORANGE);
    tegneflate.setColor(Color.BLUE);
    tegneflate.setFont(skrift);
    tegneflate.drawString("Hei hei", 50, 100);
    tegneflate.setColor(Color.RED);
    tegneflate.drawOval(40, 30, 55, 40);
  }
}
```

Oppgave 1b

```
class Vindu extends JFrame {
  public Vindu(String tittel) {
    setTitle(tittel);
    setSize(300, 200); // bredde, høyde
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    String tekst = JOptionPane.showInputDialog("Skriv teksten: ");
    Container guiBeholder = getContentPane();
    Tegning tegningen = new Tegning(tekst);
    guiBeholder.add(tegningen);
  }
}

class Tegning extends JPanel {
  private final String tekst;
  private static final Font skrift = new Font("Dialog", Font.BOLD, 16);

  public Tegning(String tekst) {
    this.tekst = tekst;
  }
  public void paintComponent(Graphics tegneflate) {
    super.paintComponent(tegneflate);  // Husk denne!
    setBackground(Color.ORANGE);
    tegneflate.setColor(Color.BLUE);
    tegneflate.setFont(skrift);
    tegneflate.drawString(tekst, 50, 100);
```

```
    tegneflate.setColor(Color.RED);
    tegneflate.drawOval(40, 30, 55, 40);
  }
}
```

Oppgave 2

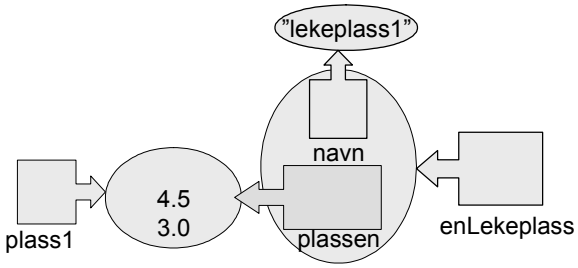
Høyden blir parameter til konstruktøren Vindu(). Det blir dermed endringer i main() og i klassen Vindu:

```
public static void main(String[] args) {
  String høydeS = showInputDialog("Oppgi skriftstørrelse: ");
  int høyde = Integer.parseInt(høydeS);
  Vindu etVindu = new Vindu("Ulike skrifttyper", høyde);
  etVindu.setVisible(true);
}

class Vindu extends JFrame {
  public Vindu(String tittel, int høyde) {  // ny parameter
    setTitle(tittel);
    setSize(500, 250); // bredde, høyde
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    Tegning tegningen = new Tegning(høyde);
    add(tegningen);
  }
}
```

Klassen Tegning endres ikke.

Kapittel 11.3

Oppgave 1



Oppgave 2

Først settes lengden lik 8.5. Deretter endres den til 5.0.

Oppgave 3

```
A: Lengde: 4.5, bredde: 3.0
B: Lengde: 5.0, bredde: 3.0
```

Kapittel 11.4

Oppgave 1

Innholdet i variablene tall1 og tall2 byttes om, slik at utskriften blir:

```
Før: 3 4
Etter: 4 3
```

Oppgave 2

Utskriften blir slik:

```
Før: 10 4
Etter: 10 4
```

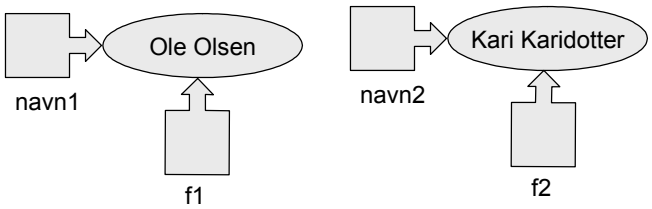
Ombytting inne i en metode har ingen effekt på variablene hos klienten, på grunn av at metoden jobber med kopier av disse variablene.

Oppgave 3

Problemet i begge tilfellene er at metodene endrer verdiene til referansene, men ikke til det som referansene opprinnelig pekte til. Vi skal gå gjennom dette litt mer detaljert. Anta at begge metodene ligger inne i en klasse som heter NavneBytter. Vi har også et enkelt klientprogram:

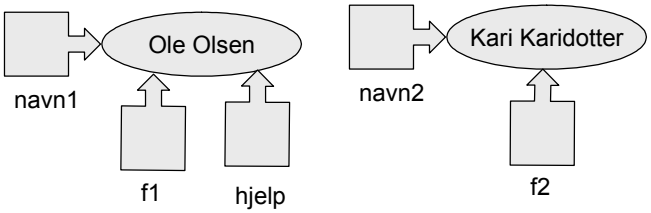
```
class Oppg11_4_3 {
  public static void main(String[] args) {
    NavneBytter ombytter = new NavneBytter();
    Navn navn1 = new Navn("Ole", "Olsen");
    Navn navn2 = new Navn("Kari", "Karidotter");
    ombytter.byttOmObjekter1(navn1, navn2);
    System.out.println("A " + navn1 + " " + navn2);
    ombytter.byttOmObjekter2(navn1, navn2);
    System.out.println("B " + navn1 + " " + navn2);
  }
}

/* Utskrift:
A Ole Olsen Kari Karidotter
B Ole Olsen Kari Karidotter
*/
```

Anta at programkontrollen står i main() ved kallet på byttOmObjekter1(). Parameter settes lik argument som vist på figuren nedenfor:

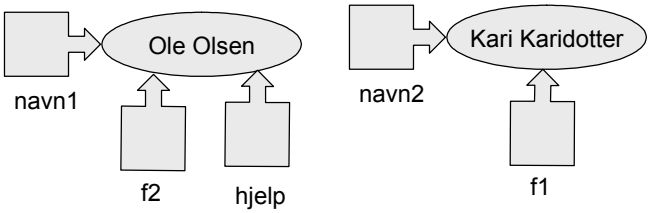


Hva skjer inne i metoden? Jo, vi oppretter en ny referanse, hjelp, som vi setter til å peke til det samme som f1 peker til:

```
Navn hjelp = f1;
```

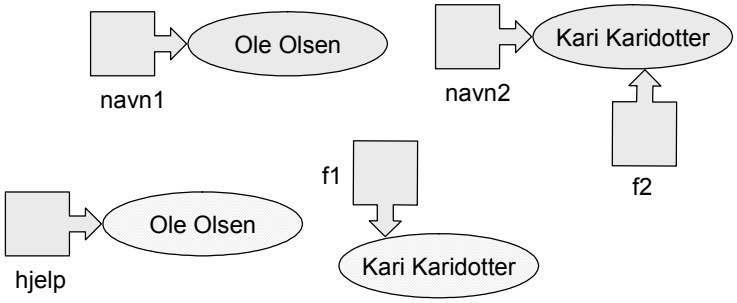


Vi bruker hjelp til å bytte om referansene slik at sluttresultatet blir som følger:



Men på grunn av at det ikke har skjedd noe med klientens referanser, navn1 og navn2, så har ikke denne ombyttingen noen effekt.

Hva med den andre metoden, byttOmObjekter2()? Argumentoverføringen er som for byttOmObjekter1(). Vi oppretter to nye objekter inne i metoden:



De to objektene er skravert på figuren. Vi ser situasjonen før siste setning i metoden utføres. Denne setningen fører til at f2 settes til å peke til det skraverte "Ole Olsen"-objektet. Igjen er problemet at vi forandrer på referansene inne i metoden, og det får ingen effekt utenfor.

Oppgave 4

Dersom vi skal endre på objekter som tilhører klienten, må vi *ikke* flytte på referansene inne i metoden:

```
public void byttOmObjekter3(Navn f1, Navn f2) {
  Navn temp = new Navn(f1.getFornavn(), f1.getEtternavn());
  f1.setFornavn(f2.getFornavn());
  f1.setEtternavn(f2.getEtternavn());
  f2.setFornavn(temp.getFornavn());
  f2.setEtternavn(temp.getEtternavn());
```

```
}
```

Her mellomlagrer vi dataene i et hjelpeobjekt, men referansene f1 og f2 endres ikke.

Følgende to setninger

```
ombytter.byttOmObjekter3(navn1, navn2);
System.out.println("C " + navn1 + " " + navn2);
```

lagt inn til slutt i klientprogrammet foran gir denne utskriften:

```
C Kari Karidotter Ole Olsen
```

Navnene er nå byttet om.

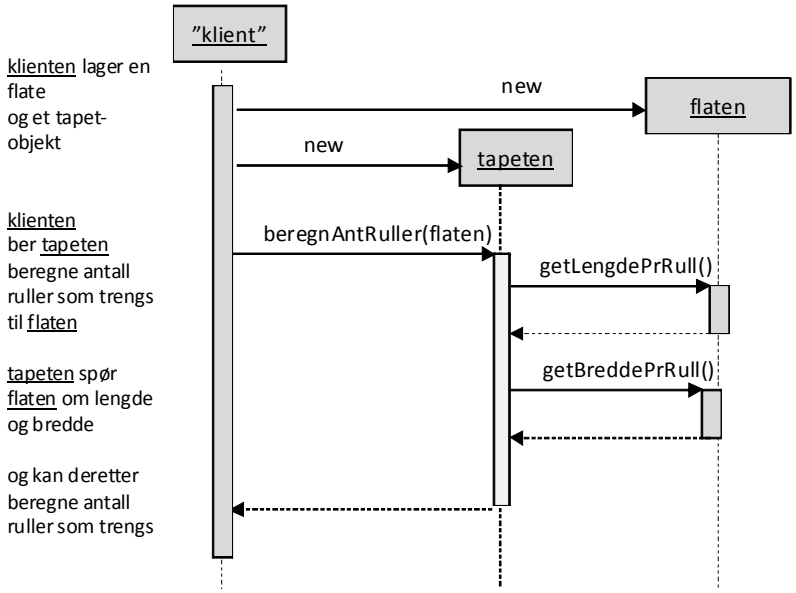
Kapittel 11.5

```
class Person {
  private final String navn;
  private final int fødselsår;
  public Person(String navn, int fødselsår) {
    this.navn = navn;
    this.fødselsår = fødselsår;
  }
  public String getNavn() {
    return navn;
  }
  public int getFødselsår() {
    return fødselsår;
  }

  public boolean likeGammel(Person p) {  // oppgave 1
    return (p.fødselsår == fødselsår);  // kan også bruke p.getFødselsår()
  }

  public int sammenliknAlder(Person p) {  // oppgave 2
    if (fødselsår < p.fødselsår) {
      return -1;
    } else if (fødselsår > p.fødselsår) {
      return +1;
    } else {
      return 0;
    }
  }

  public boolean flereÅrEldreEnn(Person p, int antÅr) {  // oppgave 3
    return (fødselsår - antÅr > p.fødselsår);
  }

  public boolean equals(Object obj) { // oppgave 4
    if (!(obj instanceof Person)) {
      return false;
    }
    if (this == obj) {
```

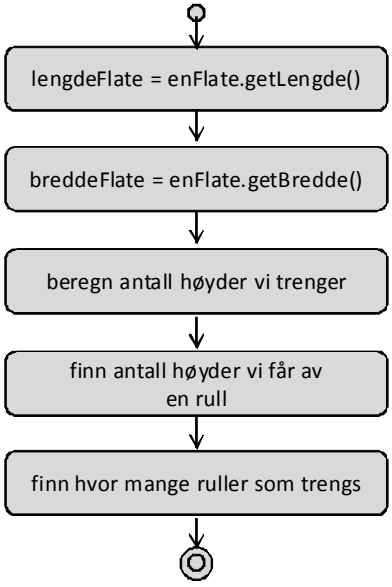
```
      return true;
    }
    Person p = (Person) obj;
    return navn.equals(p.getNavn());
  }
}

/* *
 *
 * Klientprogram
 *
 */
class Oppg11_5_1til4 {
  public static void main(String[] args) {
    Person p1 = new Person("Ole", 1980);
    Person p2 = new Person("Kari", 1984);
    if (p1.likeGammel(p2)) {
      System.out.println("Like gamle");
    } else {
      System.out.println("Ikke like gamle");
    }
    if (p1.sammenliknAlder(p2) < 0) {
      System.out.println("p1 er yngst");
    } else if (p1.sammenliknAlder(p2) > 0) {
       System.out.println("p1 er eldst");
    } else {
      System.out.println("p1 og p2 er like gamle");
    }
    if (p1.flereÅrEldreEnn(p2, 3)) {
      System.out.println("p1 er mer enn 3 år eldre enn p2");
    }

    if (p1.equals(p2)) {
      System.out.println("Her er det noe feil");
    } else {
      System.out.println("Ikke likhet.");
    }

    if (p1.equals(p1)) {
      System.out.println("Likhet");
    } else {
      System.out.println("Feil.");
    }
  }
}
/*
Ikke like gamle
p1 er yngst
Ikke likhet.
Likhet
*/
```

Kapittel 11.6

Oppgave 1a



Oppgave 1b



Oppgave 2

```
/**
 * Maling.java
```

```
 *
 * Klassen Maling tilbyr metoder for beregning av materialbehov og pris.
 * Attributter: Malingsnavn (identifikator), pris, antall strøk og dekke-evne.
 */
class Maling {
  private static final double GRENSE = 0.02;  // grense for å kjøpe 0.5 liter maling til
  private final String navn;  // identifiserer malingen
  private final double pris;  // pr.liter
  private final int antStrøk;
  private final double antKvmPrLiter;

  public Maling(String navn, double pris, int antStrøk, double antKvmPrLiter) {
    if (navn == null || navn.trim().equals("")) {
      throw new IllegalArgumentException("Malingsnavn må være oppgitt.");
    }
    if (pris <= 0.0 || antStrøk <= 0 || antKvmPrLiter <= 0.0) {
      throw new IllegalArgumentException(
              "Både pris, ant. strøk og dekkevne (kvm/l) må være positive tall.\n" +
              "Inndata til konstruktøren var: pris = " + pris +
              ", ant. strøk = " + antStrøk + ", dekkevne (kvm/l) = " + antKvmPrLiter);
    }
    this.navn = navn;
    this.antStrøk = antStrøk;
    this.antKvmPrLiter = antKvmPrLiter;
    this.pris = pris;
  }

  public String getNavn() {
    return navn;
  }

  public double getPrisPrLiter() {
    return pris;
  }

  public int getAntStrøk() {
    return antStrøk;
  }

  public double getAntKvmPrLiter() {
    return antKvmPrLiter;
  }

  /**
   * Metoden finner antall liter maling som trengs,
   * behovet rundes oppover til nærmeste halve liter.
   */
  public double beregnAntLiter(Flate enFlate) {
    double areal = enFlate.beregnAreal();
    double antLiter = areal * antStrøk / antKvmPrLiter;
    int heltAntallLitere = (int) antLiter;
```

```
    double overLiteren = antLiter - heltAntallLitere;
    if (overLiteren >= 0.5 + GRENSE) {
      return heltAntallLitere + 1.0;
    } else {
      if (overLiteren >= GRENSE) {
        return heltAntallLitere + 0.5;
      } else {
        return heltAntallLitere;
      }
    }
  }

  /**
   * Metoden regner ut hva malingen til en bestemt flate koster.
   */
  public double beregnTotalpris(Flate enFlate) {
    return beregnAntLiter(enFlate) * pris;
  }

  /**
   * To objekter er like dersom navnet er det samme.
   */
  public boolean equals(Object obj) {
    if (!(obj instanceof Maling)) {
      return false;
    }
    if (this == obj) {
      return true;
    }
    Maling m = (Maling) obj;
    return (navn.equals(m.getNavn()));
  }

  public String toString() {
    java.util.Formatter f = new java.util.Formatter();
    f.format("Maling: %s, pris kr  %.2f  pr. liter, ant. strøk: %d, ant. kvm/l %.2f",
             navn, pris, antStrøk, antKvmPrLiter);
    return f.toString();
  }
}
```

Kapittel 12.1

Oppgave 1

Oppgaven inneholder en utilsiktet feil. Metodenavnet settPris() er brukt. Det skal være setPris(). Etter at dette er rettet, vil kodebiten ved kjøring kaste to typer unntak:

- NullPointerException: Tabellen varene er en tabell av referanser. Disse referansene må settes til å peke til objekter, før vi kan sende meldinger til objektene. Med andre ord: Vi må skrive for eksempel:

```
varene[0] = new Vare("TV-dress", 100, 575.50);
```

før vi kan sende melding til dette objektet:

```
varene[1].setPris(320.50);
```

- ArrayIndexOutOfBoundException: Dette unntaket kastes når vi i siste setning refererer til tabellelement med indeks 3. Elementene i en tabell med størrelse 3 nummereres 0, 1 og 2.

Oppgave 2a

```
navneliste[1] = new String("Anders");
```

eller

```
navneliste[1] = "Anders";
```

Oppgave 2b

```
navneliste[3] = etNavn;
```

Oppgave 2c

```
int sumLengde = 0;
for (int i = 0; i < navneliste.length; i++) {
  sumLengde += navneliste[i].length();
}
```

Oppgave 2d

```
final char TEGN = 'r';
int antTegn = 0;
for (int i = 0; i < navneliste.length; i++) {
  int indeks = navneliste[i].indexOf(TEGN);
  while (indeks >= 0) {
    antTegn++;
    indeks = navneliste[i].indexOf(TEGN, indeks + 1);
  }
}
System.out.println("Antall forekomster av " + TEGN + " er: " + antTegn);
```

Oppgave 3

```
class Oppg12_2_3 {
  public static void main(String[] args) {

    //Oppgave 3a
    Vare[] varer = new Vare[4];
    varer[0] = new Vare("ost", 100, 70);
    varer[1] = new Vare("pølse", 101, 60);
    varer[2] = new Vare("banan", 102, 9.50);
    varer[3] = new Vare("epler", 103, 14.50);

    //Oppgave 3b
    for (int i = 0; i < varer.length; i++) {
      System.out.println(varer[i].getVarenavn());
     }

    //Oppgave 3c
```

```
    double maksPris = varer[0].getPris();
    int indeksDyresteVare = 0;
    for (int i = 1; i < varer.length; i++) {
      if (varer[i].getPris() > maksPris) {
        maksPris = varer[i].getPris();
        indeksDyresteVare = i;
      }
    }
    System.out.println("Dyreste vare er " + varer[indeksDyresteVare].getVarenavn()
      + ", og den koster kr. " + maksPris + " pr. kg.");

    //Oppgave 3d
    for (int i = 0; i < varer.length; i++) {
      double pris = varer[i].getPris();
      pris *= 1.07;
      varer[i].setPris(pris);
    }
  }
}
/* Utskrift:
ost
pølse
banan
epler
Dyreste vare er ost, og den koster kr. 70.0 pr. kg.
*/
```

Oppgave 4

```
public static void main(String[] args) {
  for (int i = 0; i < args.length; i++) {
    System.out.println(args[i]);
  }
}
```

Kapittel 12.2

Oppgave 1

```
/* Oppgave 1a */
public int finnTotAntStudenter() {
  int sum = 0;
  for (int i = 0; i < antFag; i++) {
    sum += fagene[i].getAntStud();
  }
  return sum;
}

/* Oppgave 1b */
public Fag[] finnStørsteFag() {
  int maks = finnMaksAntStud(); // hjelpemetode, se nedenfor
  Fag[] fagMedMaks = new Fag[antFag];
  int antFagLikMaks = 0;
  for (int i = 0; i < antFag; i++) {
```

```
      if (fagene[i].getAntStud() == maks) {
        fagMedMaks[antFagLikMaks] = fagene[i];
        antFagLikMaks++;
      }
    }
    Fag[] nyTab = new Fag[antFagLikMaks];
    for (int i = 0; i < antFagLikMaks; i++) {
      nyTab[i] = fagMedMaks[i];
    }
    return nyTab;
  }

  private int finnMaksAntStud() {
    if (antFag > 0) {
      int maks = fagene[0].getAntStud();
      for (int i = 1; i < antFag; i++) {
        if (fagene[i].getAntStud() > maks) {
          maks = fagene[i].getAntStud();
        }
      }
      return maks;
    }
    return 0; // ingen fag registrert
  }
```

Testklient:

```
  public static void main(String[] args) {
    System.out.println("Totalt antall tester: 4");

    /* Tom katalog */
    Fagkatalog kat0 = new Fagkatalog();
    Fag[] fag0 = kat0.finnStørsteFag();
    if (fag0.length == 0) {
      System.out.println("Oppgave 12-2: Test 1 vellykket");
    }
    if (kat0.finnTotAntStudenter() == 0) {
      System.out.println("Oppgave 12-2: Test 2 vellykket");
    }

    /* Katalog med 5 fag */
    Fagkatalog kat = new Fagkatalog();
    kat.registrerNyttFag("LC191D", "Videregående programmering");
    kat.registrerNyttFag("LV172D", "Programmering i Java");
    kat.registrerNyttFag("LO347D", "Web-applikasjoner");
    kat.registrerNyttFag("LO346D", "Java EE");
    kat.registrerNyttFag("LC331D", "IT, miljø og samfunn");
    kat.oppdaterAntStud("LC191D", 20);
    kat.oppdaterAntStud("LV172D", 30);
    kat.oppdaterAntStud("LO347D", 20);
    kat.oppdaterAntStud("LO346D", 30);
    kat.oppdaterAntStud("LC331D", 30);
```

```
    Fag[] fag = kat.finnStørsteFag();
    if (fag.length == 3 && fag[0].getFagkode().equals("LV172D")
                        && fag[1].getFagkode().equals("LO346D")
                         && fag[2].getFagkode().equals("LC331D")) {
      System.out.println("Oppgave 12-2: Test 3 vellykket");
    }
    if (kat.finnTotAntStudenter() == 130) {
      System.out.println("Oppgave 12-2: Test 4 vellykket");
    }
  }
```

Oppgave 2

De fem setningene med kall på oppdaterAntStud() skiftes ut med:

```
Fag f = kat.finnFagGittKode("LC191D");
f.setAntStud(20);
f = kat.finnFagGittKode("LV172D");
f.setAntStud(30);
f = kat.finnFagGittKode("LO347D");
f.setAntStud(20);
f = kat.finnFagGittKode("LO346D");
f.setAntStud(30);
f = kat.finnFagGittKode("LC331D");
f.setAntStud(30);
```

Dette er en måte å endre datainnholdet i en "katalog" på dersom katalog-klassen ikke tilbyr metoder for å gjøre dette direkte.

Kapittel 12.3

Oppgave 1

Legger kopikonstruktøren fra side 404 inn i klassen Fag, og bruker den ved kopiering.

```
class Oppg12_3_1 {
  public static void main(String[] args) {
    Fag[] gruppeA = {new Fag("LV172D", "Programmering i Java"),
                     new Fag("LC191D", "Videregående programmering"),
                     new Fag("LO347D", "Web-applikasjoner med Java EE")};

    /* Setter antall studenter til hhv 10, 15 og 20, og skriver ut */
    gruppeA[0].setAntStud(10);
    gruppeA[1].setAntStud(15);
    gruppeA[2].setAntStud(20);
    System.out.println("\nUtskrift 1 - gruppeA:");
    for (int i = 0; i < gruppeA.length; i++) {
      System.out.println(gruppeA[i]); // toString() blir brukt
    }

    /*
     * Lager gruppeB ved å kopiere gruppeA, endrer ant.stud. i gruppeB
     * og skriver ut både gruppeB og gruppeA
```

```
     */
    Fag[] gruppeB = new Fag[gruppeA.length];
    for (int i = 0; i < gruppeA.length; i++) {
      gruppeB[i] = new Fag(gruppeA[i]); // bruker kopikonstruktøren
    }
    gruppeB[0].setAntStud(20);
    gruppeB[1].setAntStud(25);
    gruppeB[2].setAntStud(30);
    System.out.println("\nUtskrift 2 - gruppe A og B:");
    for (int i = 0; i < gruppeA.length; i++) {
      System.out.println("GruppeA: " + gruppeA[i].getAntStud()
                         + ", GruppeB: " + gruppeB[i].getAntStud());
    }

    /*
     * Lager gruppeC ved å kopiere gruppeA. Samtidig endres ant.stud. i gruppeC.
     * og skriver ut både gruppeC og gruppeA
     */
    Fag[] gruppeC = new Fag[gruppeA.length];
    for (int i = 0; i < gruppeA.length; i++) {
      gruppeC[i] = new Fag(gruppeA[i]); // bruker kopikonstruktøren
      gruppeC[i].setAntStud(gruppeA[i].getAntStud() * 2);
    }
    System.out.println("\nUtskrift 3 - gruppe A og C:");
    for (int i = 0; i < gruppeA.length; i++) {
      System.out.println("GruppeA: " + gruppeA[i].getAntStud()
                         + ", GruppeC: " + gruppeC[i].getAntStud());
    }
  }
}

/* Kjøring av programmet:

Utskrift 1 - gruppeA:
Kode: LV172D, Navn: Programmering i Java, 10 studenter.
Kode: LC191D, Navn: Videregsende programmering, 15 studenter.
Kode: LO347D, Navn: Web-applikasjoner med Java EE, 20 studenter.

Utskrift 2 - gruppe A og B:
GruppeA: 10, GruppeB: 20
GruppeA: 15, GruppeB: 25
GruppeA: 20, GruppeB: 30

Utskrift 3 - gruppe A og C:
GruppeA: 10, GruppeC: 20
GruppeA: 15, GruppeC: 30
GruppeA: 20, GruppeC: 40
*/
```

Kapittel 12.4

Første del av klassen Fag der standardsortering er implementert:

```
class Fag implements Comparable<Fag> {
  private final String fagkode;  // entydig
  private final String fagnavn;
  private int antStud = 0;  // endres med metoden setAntStud()

  private final static java.text.Collator KOLLATOR = java.text.Collator.getInstance();
  public int compareTo(Fag detAndre) {
    return KOLLATOR.compare(fagkode, detAndre.fagkode);
  }
```

De to Comparator-klassene:

```
class FagKompNavn implements java.util.Comparator<Fag> {
  private final static java.text.Collator KOLLATOR = java.text.Collator.getInstance();
  public int compare(Fag fag1, Fag fag2) {
    return KOLLATOR.compare(fag1.getFagnavn(), fag2.getFagnavn());
  }
}

class FagKompAntStud implements java.util.Comparator<Fag> {
  public int compare(Fag fag1, Fag fag2) {
    if (fag1.getAntStud() < fag2.getAntStud()) {
      return -1;
    } else if (fag1.getAntStud() > fag2.getAntStud()) {
      return 1;
    } else {
      return 0;
    }
  }
}
```

Klientprogram:

```
class Oppg12_4_1 {
  public static void main(String[] args) {
    Fag f1 = new Fag("LV172D", "Programmering i Java");
    Fag f2 = new Fag("LC191D", "Videregående programmering");
    f1.setAntStud(25);
    f2.setAntStud(40);
    System.out.println(f1.compareTo(f2));  // -1
    System.out.println(f2.compareTo(f2));  // 0
    System.out.println(f2.compareTo(f1));  // 1
    System.out.println();

    FagKompAntStud komp1 = new FagKompAntStud();
    System.out.println(komp1.compare(f1, f2));  // 1
    System.out.println(komp1.compare(f2, f2));  // 0
    System.out.println(komp1.compare(f2, f1));  // -1
    System.out.println();

    FagKompNavn komp2 = new FagKompNavn();
    System.out.println(komp2.compare(f1, f2)); // 1
    System.out.println(komp2.compare(f2, f2)); // 0
    System.out.println(komp2.compare(f2, f1)); // +1
```

```
  }
}
```

Kapittel 12.5

Oppgave 1

```
class Oppg12_5_1 {
  public static void main(String[] args) {
    Fag[] fagTab = {new Fag("LV172D", "Programmering i Java"),
                    new Fag("LC191D", "Videregående programmering"),
                    new Fag("LO347D", "Web-applikasjoner med Java EE"),
                    new Fag("LC331D", "IT, miljø og samfunn"),
                    new Fag("LV299D", "IT verktøy")};

    /* Standard sortering (etter fagkode). */
    Sortering.sorterObjekter(fagTab);
    System.out.println("Sortert etter fagkode:");
    for (int i = 0; i < fagTab.length; i++) {
      System.out.println(fagTab[i]);
    }

    /* Sortering etter fagnavn. */
    Sortering.sorterObjekter(fagTab, new FagKompNavn());
    System.out.println("\nSortert etter fagnavn:");
    for (int i = 0; i < fagTab.length; i++) {
      System.out.println(fagTab[i]);
    }
  }
}
```

Kapittel 12.6

```
class Oppg12_6_1 {
  public static void main(String[] args) {
    Fag[] fagTab = {new Fag("LV172D", "Programmering i Java"),
                    new Fag("LC191D", "Videregående programmering"),
                    new Fag("LO347D", "Web-applikasjoner med Java EE"),
                    new Fag("LC331D", "IT, miljø og samfunn"),
                    new Fag("LV299D", "IT verktøy")};

    /* Standard sortering (etter fagkode). */
    java.util.Arrays.sort(fagTab);
    System.out.println("Sortert etter fagkode:");
    for (int i = 0; i < fagTab.length; i++) {
      System.out.println(fagTab[i]);
    }

    /* Binærsøk iht. standard sortering (fagkode). */
    Fag fagFins = new Fag("LC331D", "xxx");
    int res1 = java.util.Arrays.binarySearch(fagTab, fagFins); // finnes
    System.out.println("Fag med kode LC331D er funnet på indeks " + res1);
    Fag fagFinsIkke = new Fag("LC333D", "xxx");
```

```
    int res2 = java.util.Arrays.binarySearch(fagTab, fagFinsIkke); // finnes
    System.out.println("Fag med kode LC333D kan plasseres inn på indeks "
                              + (-res2 - 1));

    /* Sortering etter fagnavn. */
    FagKompNavn komp = new FagKompNavn();
    java.util.Arrays.sort(fagTab, komp);
    System.out.println("\nSortert etter fagnavn:");
    for (int i = 0; i < fagTab.length; i++) {
      System.out.println(fagTab[i]);
    }

    /* Binærsøk iht. komparator (fagnavn).*/
    fagFins = new Fag("xxx", "IT verktøy");
    res1 = java.util.Arrays.binarySearch(fagTab, fagFins, komp); // finnes
    System.out.println("Faget IT-verktøy er funnet på indeks " + res1);
    fagFinsIkke = new Fag("xxx", "xxx");
    res2 = java.util.Arrays.binarySearch(fagTab, fagFinsIkke, komp); // finnes
    System.out.println("Faget xxx kan plasseres inn på indeks " + (-res2 - 1));
  }
}
```