Løsning på småoppgaver etter hvert underkapittel

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

Tilpasset 3.utgave av boka.

Kapittel 2.1

Oppgave 2

```
class Arealberegning {
  public static void main(String[] args) {
    double lengde = 5.0; // meter
    double bredde = 2.3;  // meter
    double arealet = lengde * bredde;
    System.out.println("Arealet av rektangelet er " + arealet + " kvadratmeter");
    double omkretsen = 2 * (lengde + bredde);
    System.out.println("Omkretsen av rektangelet er " + omkretsen + " meter");
  }
}
```

Kapittel 2.3

Oppgave 1

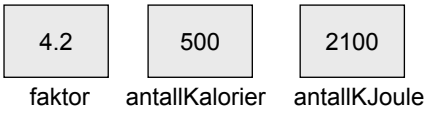
Følgende ord er lovlige navn:

```
KarisBok
evas_sykkel
AntallBær
NUMMER
_Tall34
```

Oppgave 2

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

Variabler: faktor, antallKalorier, antallKJoule.



Kapittel 2.4

Oppgave 1

Ordlyden i feilmeldingene kan variere noe fra kompilator til kompilator. Feilmeldingene nedenfor kommer fra j2sdk 1.5.0-rc.

Først kommer to meldinger:

```
double Beløp = 40,95;
```

Feil: "';' expected." Semikolon mangler i slutten av linjen *foran*.

```
final EnKonstant = 789.6;
```

Feil: "<identifier> expected." Kompilatoren forventer navnet på noe, i dette tilfellet en datatype. Setningen skal se slik ut:

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

Vi retter opp og kompilerer en gang til:

```
double Beløp = 40,95;
```

Feil: "<identifier> expected." Kompilatoren peker på 9-tallet. Problemet er desimalkommaet foran 9-tallet. Husk at Java krever punktum som desimalskille.

Vi retter opp og kompilerer for tredje gang. Også da fpr vi to feilmeldinger, den første på linjen

```
DOUBLE tall = 15.7;
```

Feil: "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 Beløp". 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 antallÅr = 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.

Kapittel 2.5

Oppgave 1

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

Oppgave 2

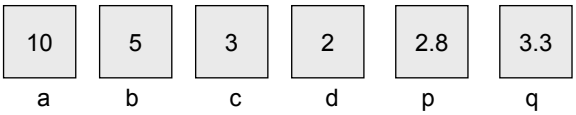
a) prisen - double
b) varemengde målt i kilo - double
c) varemengde målt i antall - int
d) fargekode (bokstav) - char
e) strekkode - String
f) navnet - String

Oppgave 3

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

Kapittel 2.6

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 ikke har egen operator for potensering.

Kapittel 2.7

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.

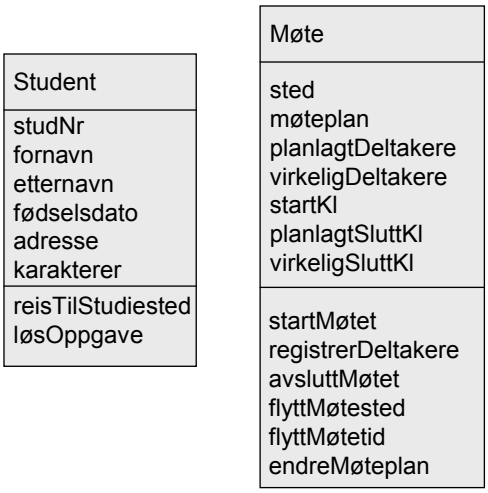
Oppgave 3

```
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



Oppgave 2

Ovnen har et volum og en maksimaleffekt. Den har gjerne funksjoner for å tine og for å koke. Effekt og koketid kan stilles inn.

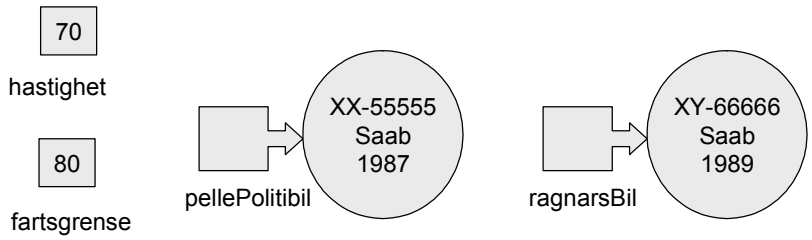
Aktuelle tilstander: Ovnen er ikke i bruk, den koker på halv effekt, ovnen tiner.

Kapittel 3.2

Oppgave 1

```
class Oppg3_2_1 {
  public static void main(String[] args) throws Exception {
    Konto karisKonto = new Konto(123456676756L, "Kari Jensen", 5000);
    karisKonto.innskudd(300.0);
    System.out.println("Etter 1. innskudd er saldoen lik " + karisKonto.finnSaldo());
    karisKonto.innskudd(1500.0);
    System.out.println("Etter 2. innskudd er saldoen lik " + karisKonto.finnSaldo());
    karisKonto.uttak(2000.0);
    System.out.println("Etter uttak er saldoen lik " + karisKonto.finnSaldo());
  }
}
/* Kjøring av programmet:
Etter 1. innskudd er saldoen lik 5300.0
Etter 2. innskudd er saldoen lik 6800.0
Etter uttak er saldoen lik 4800.0
*/
```

Oppgave 2



hastighet og fartsgrense er variabler av primitive datatyper. Variabelen fartsgrense har konstant innhold. pellePolitibil og ragnarsBil er referanser til objekter av klassen Bil. Vi antar at bilobjektene har attributtene registreringsnummer, merke og årsmodell. Objektene er tegnet med sirkler på figuren.

Oppgave 3

```
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.finnKontonr();
```

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.finnSaldo();
```

Metoden finnSaldo() tar ingen argumenter.

```
arnesKonto.innskudd(1000);
```

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

```
arnesKonto.uttak(800);
```

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

Oppgave 4

```
Student ole = new Student(56789, “Ole Jensen”);
ole.gjørEksamensoppgave(“LO-189D”);
ole.gjørEksamensoppgave(“LO-190D”);
ole.gjørEksamensoppgave(“LO-191D”);
ole.settKarakter(2.0, “LO-189D”);
ole.settKarakter(1.8, “LO-190D”);
ole.settKarakter(2.3, “LO-191D”);
ole.søkOmStudielån();
```

Kapittel 3.3

Oppgave 1

```
import java.util.Random;
class Oppg3_3_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);
  }
}
/* Kjøring:
Tre tilfeldige tall, long: -7641749851956408558, 791427278084326421,
-2678922941386077 68
Tre tilfeldig heltall [0..1000]: 67, 17, 857
To tilfeldige desimaltall [0.0..1.0>: 0.5301675050384417, 0.07132528100206137
To tilfeldige heltall [-500..500]: -390, 409
To tilfeldige desimaltall [0.0..100.0>: 69.40152954175088, 45.67180893902937
*/
```

Oppgave 2

```
import java.util.Random;
class Oppg3_3_2 {
  public static void main(String[] args) {
    Random randomGen = new Random();
    for (int i = 0; i < 20; i++) {
      System.out.println("Et tilfeldig tall: " + randomGen.nextInt());
    }
  }
}
```

Kapittel 3.5

Oppgave 1

```
class Oppg3_5_1 {
  public static void main(String[] args) {
    String tekst1 = "-4523";
    String tekst2 = "345";
    String tekst3 = "3.256e-5";

    int tall1 = Integer.parseInt(tekst1);
    int tall2 = Integer.parseInt(tekst2);
    double tall3 = Double.parseDouble(tekst3);

    System.out.println("Tall 1: " + tall1);
    System.out.println("Tall 2: " + tall2);
    System.out.println("Tall 3: " + tall3);
    System.out.println("Summen er " + (tall1 + tall2 + tall3));

    System.out.println("e: " + Math.E);

    System.out.println("Kvadratrot: " + Math.sqrt(6785.56));
  }
}

/* Utskrift:
```

```
Tall 1: -4523
Tall 2: 345
Tall 3: 3.256E-5
Summen er -4177.99996744
e: 2.718281828459045
Kvadratrot: 82.37451062070112
*/
```

Kapittel 3.7

Oppgave 1

```
import static javax.swing.JOptionPane.*;
class Oppg3_7_1 {
  public static void main(String[] args) {
    /* Den enkleste showInputDialog() metoden */
    String heltallLest = showInputDialog("Skriv et heltall: ");
    int heltall = Integer.parseInt(heltallLest);
    String desimaltallLest = showInputDialog("Skriv et desimaltall: ");
    double desimaltall = Double.parseDouble(desimaltallLest);
    String tekst = showInputDialog("Skriv en tekst: ");
    showMessageDialog(null, "Følgende er lest:\n" +
                            "Heltall: " + heltall + "\n" +
                            "Desimaltall: " + desimaltall + "\n" +
                            "Tekst: " + tekst);

    /* The mest kompliserte showInputDialog() metoden */
    heltallLest = showInputDialog(null, "Skriv et heltall: ",
       "showInputDialog() med fire parametre", QUESTION_MESSAGE);
    heltall = Integer.parseInt(heltallLest);
    desimaltallLest = showInputDialog(null, "Skriv et desimaltall: ",
       "showInputDialog() med fire parametre", QUESTION_MESSAGE);
    desimaltall = Double.parseDouble(desimaltallLest);
    tekst = showInputDialog(null, "Skriv en tekst: ",
       "showInputDialog() med fire parametre", QUESTION_MESSAGE);
    showMessageDialog(null, "Følgende er lest:\n" +
                            "Heltall: " + heltall + "\n" +
                            "Desimaltall: " + desimaltall + "\n" +
                            "Tekst: " + tekst);
  }
}
```

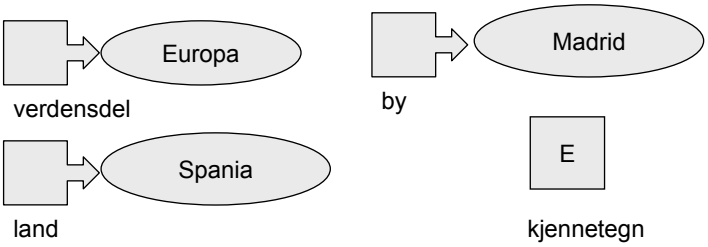
Oppgave 2

```
import static javax.swing.JOptionPane.*;
class Oppg3_7_2 {
  public static void main(String[] args) {
    final double faktor = 4.2;
    String kalLest = showInputDialog("Oppgi antall kalorier: ");
    double antallKalorier = Double.parseDouble(kalLest);
    double antallKJoule = antallKalorier * faktor;
    showMessageDialog(null, "Antall kJ blir " + antallKJoule);
```

```
    }
  }
```

Kapittel 3.8

Oppgave 1



verdensdel, land og by er referanser til objekter av klassen String. kjennetegn er en variabel av datatypen char. char er en primitiv datatype, mens String er en referansetype.

Oppgave 2

```
yesterdayy9
```

Oppgave 3

```
String tekst = "Anna Lovinda";  // ikke apostrofer, men anførselstegn
String artist = new String("Erik Bye"); // klassenavn etter new
int antTegn = artist.length( ); // parentes etter length
char sisteBokstav = artist.charAt(antTegn - 1); // tegnene nummereres fra og med 0
System.out.println("Antall tegn i artistnavnet er: " + antTegn
                        + " Siste bokstav er " + sisteBokstav);
 artist = artist.toUpperCase(); // metoden endrer ikke på objektet, men lager et nytt.
                                // Vi må derfor ta i mot referansen til det nye objektet.
System.out.println("Artistnavn med store bokstaver: " + artist);
```

Oppgave 4

```
29
6
27
Dette er en av flere oppgaver.
Dette er en av flere oppgaver.
Datta ar an av flara oppgavar.
```

Oppgave 5

```
int pos1 = tekst.indexOf('.');
int pos2 = tekst.indexOf('.', pos1 + 1);
int pos3 = tekst.indexOf('9');
int pos4 = tekst.indexOf("Om");
```

pos1 har verdien 8, pos2 er lik 21, pos3 er lik 7 og pos4 er lik 23. Neste "Om" bør kunne finnes med følgende setning:

```
int pos5 = tekst.indexOf("Om", pos4 + 1);
```

Utskrift viser at pos5 blir lik -1, som er den verdien indexOf() returnerer dersom den ikke finner det den leter etter.

Kapittel 3.9

Oppgave 1

```
import java.text.*;  // DecimalFormat og NumberFormat
import java.util.Locale;
class Oppg3_9_1 {
  public static void main(String[] args) {
    Locale tysk = new Locale("de", "DE");
    Locale.setDefault(tysk);

    double beløp = 3.467;
    DecimalFormat desFormat = new DecimalFormat("#.#");
    NumberFormat myntFormat = NumberFormat.getCurrencyInstance();

    String utskrift = "Med en desimal: " + desFormat.format(beløp) +
      ", i euro " + myntFormat.format(beløp);
    javax.swing.JOptionPane.showMessageDialog(null, utskrift);
    System.out.println(utskrift);
  }
}
/* Utskrift i MS-DOS (euro-tegnet er ikke riktig)
Med en desimal: 3,5, i euro 3,47 Ç
*/
```

Euro-tegnet vises korrekt i Windows-meldingsboksen. Det er ikke alle skrivere som klarer å skrive det ut riktig.

Oppgave 2

```
import java.text.DecimalFormat;
class Oppg3_9_2 {
  public static void main(String[] args) {
    double tall = 66779.58067;
    DecimalFormat formatA = new DecimalFormat("000000.000000");
    DecimalFormat formatB = new DecimalFormat("###,000.000");
    DecimalFormat formatC = new DecimalFormat("#.#");
    DecimalFormat formatD = new DecimalFormat("#");
    javax.swing.JOptionPane.showMessageDialog(null,
      formatA.format(tall) + "\n" + formatB.format(tall) + "\n" +
      formatC.format(tall) + "\n" + formatD.format(tall));
  }
}
```

Kapittel 4.1

Oppgave 1

```
import static javax.swing.JOptionPane.*;
class Konto {
....
class Oppg4_1_1 {
```

```
    public static void main(String[] args) throws Exception {

      /* Trinn 1: Lager et objekt av klassen */
      /* Kontonummeret er long, derfor L etter tallet */
      String kontonrLest = showInputDialog("Oppgi kontonummer: ");
      long kontonr = Long.parseLong(kontonrLest);
      String navnLest = showInputDialog("Oppgi navn: ");
      String saldoLest = showInputDialog("Oppgi startsaldo: ");
      double saldo = Double.parseDouble(saldoLest);
      Konto olesKonto = new Konto(kontonr, navnLest, saldo);

      /* Trinn 2: Prøver ut tilgangsmetodene */
      kontonr = olesKonto.finnKontonr();
      String navn = olesKonto.finnNavn();
      saldo = olesKonto.finnSaldo();
      System.out.println("Før endring av dataene: \nKontonr: " + kontonr +
        ", navn: " + navn + ", saldo: " + saldo);

      /* Trinn 3: Prøver mutasjonsmetodene */
      String innskuddLest = showInputDialog("Innskudd: ");
      double innskudd = Double.parseDouble(innskuddLest);
      olesKonto.innskudd(innskudd);

      String uttakLest = showInputDialog("Uttak: ");
      double uttak = Double.parseDouble(uttakLest);
      olesKonto.uttak(uttak);

      kontonr = olesKonto.finnKontonr();
      navn = olesKonto.finnNavn();
      saldo = olesKonto.finnSaldo();
      showMessageDialog(null, "Etter endring av dataene: \nKontonr: " +
                kontonr + ", navn: " + navn + ", saldo: " + saldo);
    }
  }
```

Oppgave 2

a) Per Ås er en bestemt person, og modelleres derfor som et objekt. En klasse er en beskrivelse av en rekke objekter.

b) Setningen utgjør et metodehode. I parentesen skal vi ha parametere, det vil si beskrivelser av argumentene. "Januar" er et argument. Beskrivelsen vil være for eksempel String måned.

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, men det må stå noe foran metodenavnet, enten void eller en datatype. Hittil har vi også hatt modifikatoren public foran. Dette kan utelates. Hva det innebærer, skal vi se på i neste underkapittel. Metodehodet kan for eksempel se slik ut: public double finnLønn(String måned)

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 blokka der den er deklarert. En objektvariabel, derimot, kan nås i alle metoder i den klassen der den er deklarert.

Oppgave 3

```
class Sirkel {
  private double radius;  // datatype mangler
  public Sirkel(double startRadius)  { // stor S
    radius = startRadius;
  } // sluttklamme mangler

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

  public double finnOmkrets() {
    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.finnAreal();
    System.out.println("Arealet er lik " + arealet);
    double omkretsen = enSirkel.finnOmkrets();
    System.out.println("Omkretsen er lik " + omkretsen);
  }
}
```

Oppgave 5

Lokale variabler: Det er bare main() som har lokale variabler: olesKonto, kontonr, navn, saldo.

Objektvariabler: Disse finner vi i klassen Konto: kontonr, navn, saldo.

parametere:

I konstruktøren: startKontonr, startNavn, startSaldo

I innskudd(): beløp

I uttak(): beløp

I main(): args

Kapittel 4.3

Oppgave 1

Dersom vi ikke lager konstruktør, lages en standardkonstruktør automatisk. Vi kan lage et objekt av klassen på følgende måte:

```
Konto enKonto = new Konto();
```

Objektvariablene kontonr, navn og saldo vil nå få de verdiene som er satt opp i deklarasjonen. Det er ikke satt opp noen spesielle verdier der, og de får derfor verdiene (i rekkefølge): 0L, null og 0.0.

Oppgave 2

Vi har samme navn på parameteren som på objektvariabelen. Dette forsøker vi å håndtere ved å bruke this. Vi har imidlertid satt this på feil side av tilordningstegnet. Dersom vi skal ha samme navn på parameter og objektvariabel, må metoden se slik ut:

```
public void settSaldo(double saldo) {
  this.saldo = saldo;
}
```

Oppgave 3a

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

I klientprogrammet:

```
System.out.println("Varenummeret er: " + enVare.finnVarenummer());
```

Oppgave 3b

```
public String finnVarenavn() {
  return varenavn;
}
```

I klientprogrammet:

```
System.out.println("Varenavnet er: " + enVare.finnVarenavn());
```

Oppgave 3c

Det er verdiene som settes i konstruktøren som gjelder.

Oppgave 3d

```
class Vare {
  private int beholdning;
   ....
  /*
   * Denne konstruktøren kommer i stedet for eller i tillegg til den
```

```
     * konstruktøren vi har fra før.
     */
    public Vare(String startVarenavn, int startVarenr, double startPris,
                                                        int startVarebeholdning) {
      varenavn = startVarenavn;
      varenr = startVarenr;
      pris = startPris;
      beholdning = startVarebeholdning;
    }

    public int finnBeholdning() {
      return beholdning;
    }
    public void økBeholdning(int antall) {
      beholdning = beholdning + antall;
    }
    public void minskBeholdning(int antall) {
      beholdning = beholdning - antall;
    }
    ....
  }
```

I klientprogrammet:

```
Vare enVare = new Vare("Norvegia", 123, kilopris1, 100, 20);
enVare.økBeholdning(30);
System.out.println("Nå er beholdningen " + enVare.finnBeholdning());
enVare.minskBeholdning(10);
System.out.println("Nå er beholdningen " + enVare.finnBeholdning());
```

Oppgave 4a

Klientprogrammet kan se slik ut:

```
public static void main(String[] args) {
  Rom romA = new Rom("A", 20);
  Rom romB = new Rom("B", 100);
  Rom romC = new Rom("C", 50);

  System.out.println("Rom A har nr. " + romA.finnRomnr());
  System.out.println("Rom B har nr. " + romB.finnRomnr());
  System.out.println("Rom C har nr. " + romC.finnRomnr());
}
```

Utskriften blir som følger:

```
Rom A har nr. 1
Rom B har nr. 2
Rom C har nr. 3
```

Oppgave 4b

Metoden ser slik ut:

```
public static int finnSistBrukteRomnr() {
  return sistBrukteRomnr;
}
```

Den kalles slik:

```
int siste = Rom.finnSistBrukteRomnr();
```

Oppgave 4c

En klassemetode kan kun referere til klassevariabler og -konstanter. Årsaken til dette er at metoden kalles på vegne av klassen, og det er dermed ikke knyttet noen bestemte objektvariabler til den.

En objektmetode kan referere til alle typer variabler.

Kapittel 4.4

Oppgave 1

```
class Vare {
  .........
  public String toString() {
    java.text.NumberFormat pengeformat =
                                        java.text.NumberFormat.getCurrencyInstance();
    return "Varenavn: " + varenavn + ", nr.: " + varenr +
                    ", pris: " + pengeformat.format(pris);
  }
}

class Oppg4_4_1 {
  public static void main(String[] args) {
    final double kilopris1 = 73.50;
    Vare enVare = new Vare("Norvegia", 124, kilopris1);
    System.out.println(enVare);
  }
}

/* Kjøring av programmet:
Varenavn: Norvegia, nr.: 124, pris: kr 73,50
*/
```

Kapittel 4.5

Oppgave 1

```
class Oppg4_5_1 {
  public static void main(String[] args) {
    JaNeiTeller tellemaskin = new JaNeiTeller();
    tellemaskin.økAntallJa();
    tellemaskin.økAntallNei();
    System.out.println("Antall ja: " + tellemaskin.finnAntallJa() +
      " Antall nei: " + tellemaskin.finnAntallNei());
    tellemaskin.økAntallJa(10);
    tellemaskin.økAntallNei(20);
    System.out.println(tellemaskin); // prøver toString()
  }
}

/* Utskrift:
```

```
Antall ja: 1 Antall nei: 1
Antall ja: 11 Antall nei: 21
*/
```

Oppgave 2

```
2
2
```

Oppgave 3

a = 121
b = 20
c = 1

Kapittel 4.7

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 Oppg4_7_1 {
  public static void main(String[] args) {
    Vindu etVindu = new Vindu("Oppgave 4-7-1");
    etVindu.setVisible(true);
  }
}
```

Kapittel 4.8

Oppgave 3a

```
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.green);
    tegneflate.setColor(Color.blue);
    tegneflate.setFont(skrift);
    tegneflate.drawString("Hei hei", 50, 100);
    tegneflate.setColor(Color.red);
    tegneflate.drawOval(40, 30, 55, 40);
  }
}
```

Oppgave 3b

```
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 String tekst;
  private static final Font skrift = new Font("Dialog", Font.BOLD, 16);

  public Tegning(String startTekst) {
    tekst = startTekst;
  }
  public void paintComponent(Graphics tegneflate) {
    super.paintComponent(tegneflate);  // Husk denne!
    setBackground(Color.green);
    tegneflate.setColor(Color.blue);
    tegneflate.setFont(skrift);
    tegneflate.drawString(tekst, 50, 100);
    tegneflate.setColor(Color.red);
    tegneflate.drawOval(40, 30, 55, 40);
  }
}
```

Kapittel 4.9

Oppgave 1

```
/* toString()-metode som erstatter den som er arvet */
public String toString() {
  String resultat = super.toString();
  if (på) resultat += " er på.";
  else resultat += " er av.";
  return resultat;
}
```

Klientprogram:

```
public static void main(String[] args) {
  System.out.println(Trafikklys.rød + " " + Trafikklys.gul + " " + Trafikklys.grønn);
}
```

Oppgave 2

```
/* Klassemetode som gjør noe med alle objektene */
public static void settPå(Trafikklys lys ) {
  if (lys == rød) {
    rød.settPå(true);
    gul.settPå(false);
    grønn.settPå(false);
  } else if (lys == gul) {
    rød.settPå(false);
    gul.settPå(true);
    grønn.settPå(false);
  } else {  // grønn
    rød.settPå(false);
    gul.settPå(false);
    grønn.settPå(true);
  }
}
```

Klientprogrammet ser nå slik ut:

```
public static void main(String[] args) {
  System.out.println(Trafikklys.rød + " " + Trafikklys.gul + " " + Trafikklys.grønn);
  Trafikklys.settPå(Trafikklys.gul);
  System.out.println("Gul på: " + Trafikklys.rød + " " + Trafikklys.gul +
      " " + Trafikklys.grønn);
  Trafikklys.settPå(Trafikklys.grønn);
  System.out.println("Grønn på: " + Trafikklys.rød + " " + Trafikklys.gul +
     " " + Trafikklys.grønn);
 }
```

Kapittel 5.1

Oppgave 1

```
class Oppg5_1_1 {
  public static void main(String[] args) {
    double tall1 = 0;  // må byttes ut for hvert datasett
    double tall2 = 0;  // må byttes ut for hvert datasett
    Kalkulator kalk = new Kalkulator(tall1, tall2);
    System.out.println("Tester med tallene " + tall1 + " og " + tall2);
    System.out.println("Summen er: " + kalk.beregnSum());
    System.out.println("Differensen er: " + kalk.beregnDifferanse());
    System.out.println("Produktet er: " + kalk.beregnProdukt());
    System.out.println("Kvotienten er: " + kalk.beregnKvotient());
    kalk.settTall(17, 20);
    System.out.println("De nye tallene er " + kalk.finnTall1() +
      " og " + kalk.finnTall2());
  }
}
```

Oppgave 2

Metoden:

```
public double løsnEnGrad() {
   return -tall2 / tall1;
  }
```

Metodeanrop i testprogrammet:

```
System.out.println("Løsningen til 1.gradslikningen er " + kalk.løsnEnGrad());
```

Oppgave 3

Legger inn en objektvariabel til:

```
private double tall3 = 0;
```

Lager en konstruktør som tar tre argumenter:

```
public Kalkulator(double startTall1, double startTall2, double startTall3) {
  tall1 = startTall1;
  tall2 = startTall2;
  tall3 = startTall3;
}
```

To metoder som beregner røttene (vi forutsetter at røttene eksisterer):

```
public double beregnRot1() {
  double diskriminant = tall2 * tall2 - 4 * tall1 * tall3;
  return (-tall2 + Math.sqrt(diskriminant)) / (2 * tall1);
}

public double beregnRot2() {
  double diskriminant = tall2 * tall2 - 4 * tall1 * tall3;
  return (-tall2 - Math.sqrt(diskriminant)) / (2 * tall1);
}
```

I klientprogrammet:

```
Kalkulator kalk = new Kalkulator(tall1, tall2, tall3);
System.out.println("Røttene i 2.gradslikningen er " +
    kalk.beregnRot1() + " og " + kalk.beregnRot2());
```

Kapittel 5.2

Oppgave 1

```
int svar = showConfirmDialog(null, "Skal tallene ganges med hverandre? ",
                             "Kalkulator", YES_NO_OPTION);
Kalkulator kalkus = new Kalkulator(tall1, tall2);
double beregnetSvar;
char operator;
if (svar == YES_OPTION) { // Yes er trykket
  beregnetSvar = kalkus.beregnProdukt();
  operator = '*';
} else { // No eller Esc er trykket, eller dialogen er lukket
  beregnetSvar = kalkus.beregnKvotient();
  operator = '/';
}
```

Kapittel 5.3

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:

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

Deklarasjonene i blokk 2:

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

Deklarasjonen i blokk 3:

Definisjonsområdet 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 5.4

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

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

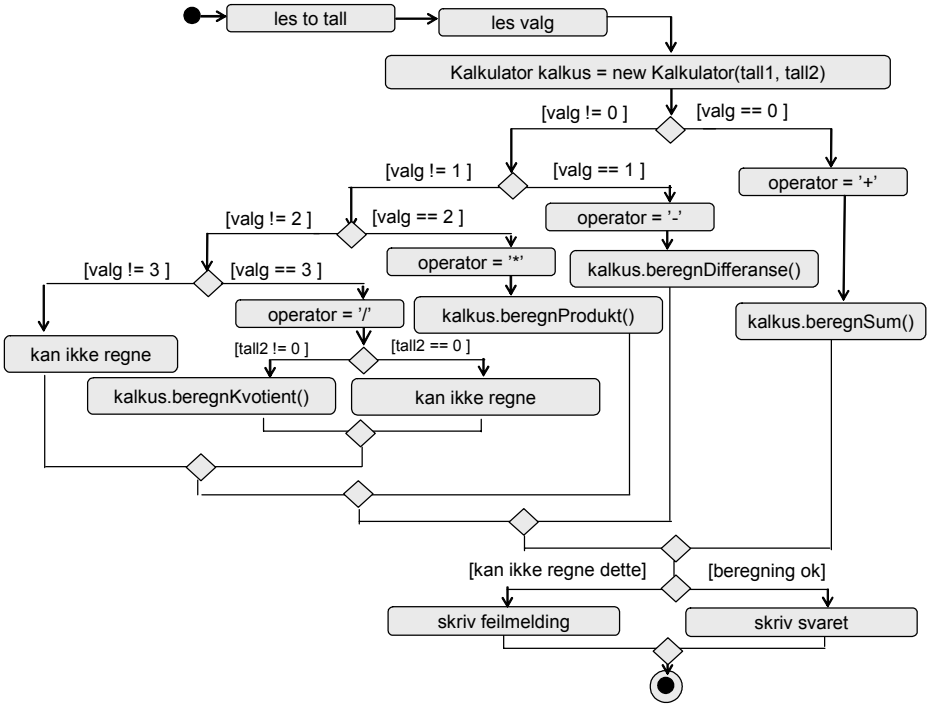
Oppgave 3

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!");
```

Kapittel 5.5

Oppgave 1



Oppgave 2

```
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 3

```
public char finnKarakter() {
  if (poeng < 0) return 'X';
  else if (poeng < 36) return 'F';
  else if (poeng < 55) return 'E';
  else if (poeng < 71) return 'D';
  else if (poeng < 86) return 'C';
  else if (poeng < 96) return 'B';
```

```
    else if (poeng < 101) return 'A';
    else return 'Y';
  }
```

Oppgave 4

```
  import static javax.swing.JOptionPane.*;
  class Eksamensresultat {
    private int poeng;
    public Eksamensresultat(int startPoeng) {
      poeng = startPoeng;
    }
    public int finnPoeng() {
      return poeng;
    }
    public char finnKarakter() {
      if (poeng > 100) return 'Y';
      else if (poeng >= 96) return 'A';
      else if (poeng >= 86) return 'B';
      else if (poeng >= 71) return 'C';
      else if (poeng >= 55) return 'D';
      else if (poeng >= 35) return 'E';
      else if (poeng >= 0) return 'F';
      else return 'X';
    }
  }
  class Oppg5_5_4 {
    public static void main(String[] args) {
      String poengS = showInputDialog(null, "Antall poeng: ");
      int poeng = Integer.parseInt(poengS);
      Eksamensresultat resultat = new Eksamensresultat(poeng);
      char karakter = resultat.finnKarakter();
      if (karakter == 'X') {
          showMessageDialog(null, "Negativ poengsum er ikke tillatt.");
      } else if (karakter == 'Y') {
         showMessageDialog(null, "Poengsummen kan ikke være over 100.");
      } else showMessageDialog(null, "Karakteren blir " + karakter);
    }
  }
```

Kapittel 5.6

Oppgave 1

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

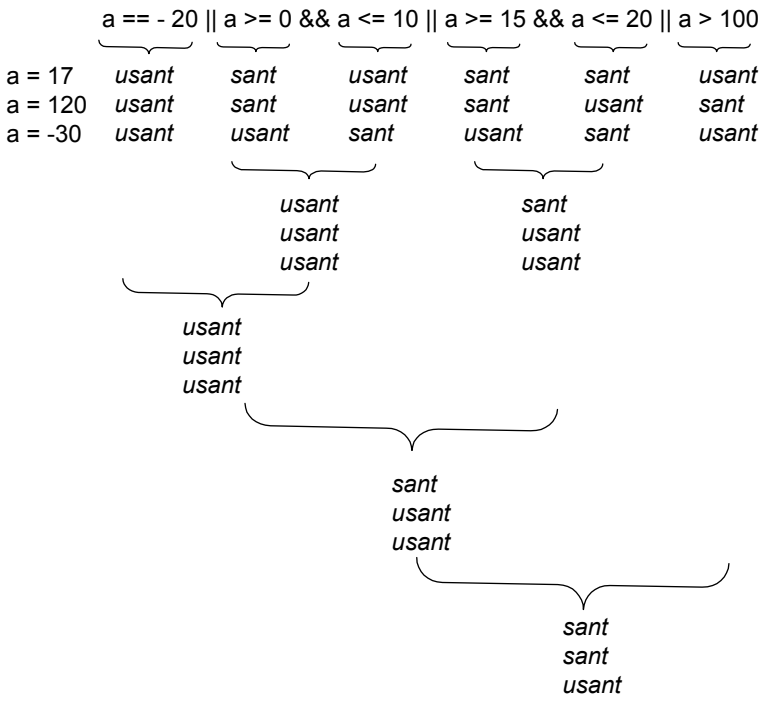
Oppgave 2

```
  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 5.7

Oppgave 1

```
false
false
true
32  (eller et annet positivt tall)
0
true
false
true  (NB! Her ser vi at kompilatoren er litt
"intelligent")
```

Kapittel 5.8

Oppgave 1

```
class Konto {
  private long   kontonr;
  private String navn;
  private long saldo;  // øre

  public Konto(long startKontonr, String startNavn, double startSaldo) {
    kontonr = startKontonr;
    navn = startNavn;
    saldo = (long) (startSaldo * 100);  // gjør om fra kroner til øre
  }

  public long finnKontonr() {
    return kontonr;
  }

  public String finnNavn() {
    return navn;
  }

  public double finnSaldo() {
    return saldo / 100.0;  // passer på å unngå heltallsdivisjon
  }

  public void innskudd(double beløp) {
    long øre = (long) (beløp * 100);
    saldo = saldo + øre;
  }

  public void uttak(double beløp) {
    long øre = (long) (beløp * 100);
    saldo = saldo - øre;
  }
}
```

Kapittel 5.9

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.

Oppgave 2

```
enum Ukedag {
  søndag, mandag, tirsdag, onsdag, torsdag, fredag, lørdag;
}

class Oppg5_9_2 {
  public static void main(String[] args) {
    Ukedag ukedag = Ukedag.torsdag;
    switch (ukedag) {
      case mandag:
        /* ikke break */
      case tirsdag:
        System.out.println("Begynnelsen av uka");
        break;
      case onsdag:
        /* ikke break */
      case torsdag:
        System.out.println("Midt i uka");
        break;
      case fredag:
        System.out.println("Slutten av uka");
        break;
      case lørdag:
        /* ikke break */
      case søndag:
        System.out.println("Helg");
        break;
      default:
        System.out.println("Ugyldig ukedag: " + ukedag);
        break;
    }
  }
}
```

Kapittel 6.1

Oppgave 1

Null ganger:

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

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++;
}
```

Kapittel 6.2

Oppgave 1

```
import java.util.Random;
class Oppg6_2_1 {
  public static void main(String[] args) {
    Random tallGen = new Random();
    int småTall = 0;
    int storeTall = 0;
    int tall = tallGen.nextInt(500) + 1;
    while (tall != 250) {
      System.out.println("Tall funnet: " + tall);
      if (tall > 250) storeTall++;
      else småTall++;
      tall = tallGen.nextInt(500) + 1;
    }
    System.out.println("Antall store tall: " + storeTall);
    System.out.println("Antall små tall: " + småTall);
  }
}
```

Oppgave 2

```
import static javax.swing.JOptionPane.*;
class Oppg6_2_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) {
```

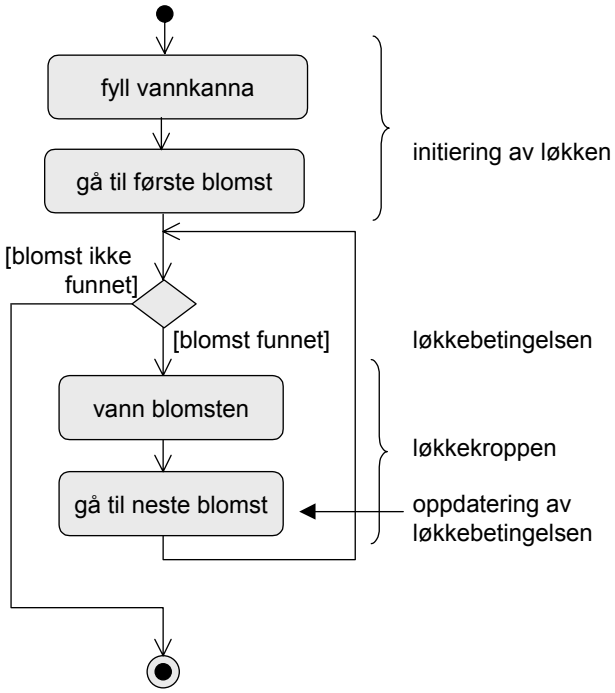
```
            svar *= x; // svaret skal ganges med x, ikke med n
            teller++; // telleren skal økes, ikke n
          }
          showMessageDialog(null, "Svaret er " + svar);
        }
    }
```

Oppgave 3

Programmet fungerer ikke for negativ eksponent.

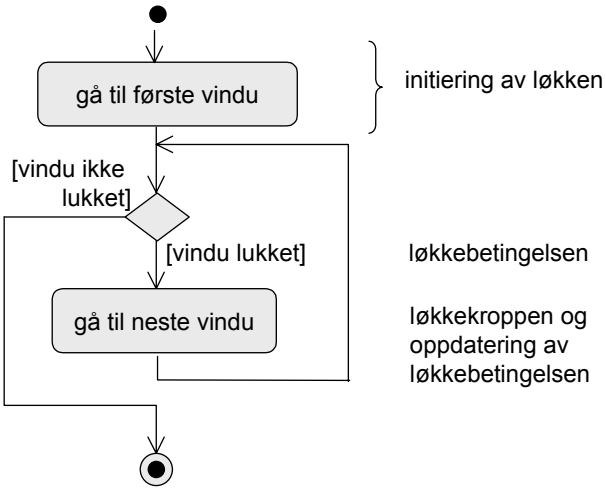
datasett nr.	n	forv. svar med grunntall -2	forv. svar med grunntall 0	forv. svar med grunntall +2
1	0	1	1	1
2	1	-2	0	2
3	3	-8	0	8

Oppgave 4a

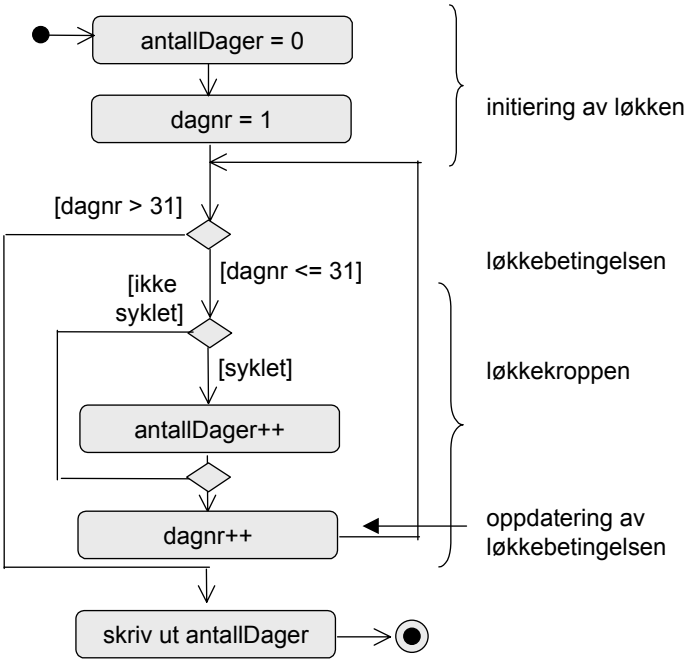


Vi forutsetter at det er nok å fylle vannkanna en gang. Hvis det ikke er det, får vi en test på om det er nok vann inne i løkka. Denne testen må ligge foran aktiviteten "vann blomsten". Hvis det ikke er nok vann, må vi fylle kanna før vi vanner.

Oppgave 4b



Oppgave 4c



Kapittel 6.3

Oppgave 1

```
public void paintComponent(Graphics tegneflate) {
```

```
    super.paintComponent(tegneflate);
    maksXverdi = getWidth() - 1;
    maksYverdi = getHeight() - 1;

    int x = tallGen.nextInt(maksXverdi - diameter);
    int y = tallGen.nextInt(maksYverdi - diameter);
    int x1 = x;  // Tar vare på koordinatene til det første punktet.
    int y1 = y;
    tegneflate.fillOval(x, y, diameter, diameter);
    int teller = 0;
    while (teller < antStreker) {
      int forrigeX = x;
      int forrigeY = y;
      x = tallGen.nextInt(maksXverdi - diameter);
      y = tallGen.nextInt(maksYverdi - diameter);
      tegneflate.drawLine(forrigeX + sentr, forrigeY + sentr, x + sentr, y + sentr);
      tegneflate.fillOval(x, y, diameter, diameter);
      teller++;
    }
    /* Tegner en linje mellom siste og første sirkel */
    tegneflate.drawLine(x + sentr, y + sentr, x1 + sentr, y1 + sentr);
  }
```

Oppgave 2

En logiske variabel, fylt, styrer hvorvidt sirkelen skal være fylt eller ikke. Legg merke til hvordan vi skifter verdi på den logiske variabelen før hver tegning.

```
  public void paintComponent(Graphics tegneflate) {
    super.paintComponent(tegneflate);
    maksXverdi = getWidth() - 1;
    maksYverdi = getHeight() - 1;
    boolean fylt = true;

    int x = tallGen.nextInt(maksXverdi - diameter);
    int y = tallGen.nextInt(maksYverdi - diameter);
    tegneflate.fillOval(x, y, diameter, diameter);
    int teller = 0;
    while (teller < antStreker) {
      int forrigeX = x;
      int forrigeY = y;
      x = tallGen.nextInt(maksXverdi - diameter);
      y = tallGen.nextInt(maksYverdi - diameter);
      tegneflate.drawLine(forrigeX + sentr, forrigeY + sentr, x + sentr, y + sentr);
      if (fylt) tegneflate.fillOval(x, y, diameter, diameter);
      else tegneflate.drawOval(x, y, diameter, diameter);
      fylt = !fylt;
      teller++;
    }
  }
```

Kapittel 6.4

Oppgave 1

```
public void paintComponent(Graphics tegneflate) {
  super.paintComponent(tegneflate);
  int x = 0;
  int y = 50;
  int diameter = 20;
  boolean fylt = true;
  for (int teller = 0; teller < 30; teller++) {
    if (fylt) tegneflate.fillOval(x, y, diameter, diameter);
    else tegneflate.drawOval(x, y, diameter, diameter);
    fylt = !fylt;
    x += diameter;
    diameter += 2;
  }
}
```

Kapittel 6.5

Oppgave 1

ABVVUIHJV

gir utskriften

```
VV
VVVV
VVVVVVVV
```

Programmet leter etter V'er i tekst. Den første V'en fører til at to V'er skrives ut, den andre til at fire V'er skrives ut, den tredje til at åtte V'er skrives ut, osv.

Oppgave 2

```
class Tegning extends JPanel {
  private static final int startDiameter = 6;
  private static final int diameterInkr = 4;
  private static final int antStreker = 10;
  private static final int fradrag = startDiameter + diameterInkr * antStreker;

  private static Random tallGen = new Random();

  private int maksXverdi;
  private int maksYverdi;

  public void paintComponent(Graphics tegneflate) {
    super.paintComponent(tegneflate);
    maksXverdi = getWidth() - 1;
    maksYverdi = getHeight() - 1;

    int x = tallGen.nextInt(maksXverdi - fradrag) + fradrag / 2;
    int y = tallGen.nextInt(maksYverdi - fradrag) + fradrag / 2;
    tegneflate.fillOval(x, y, startDiameter, startDiameter);
    int antSirkler = 1;
    int teller = 0;
    while (teller < antStreker) {
      int forrigeX = x;
```

```
        int forrigeY = y;
        x = tallGen.nextInt(maksXverdi - fradrag) + fradrag / 2;
        y = tallGen.nextInt(maksYverdi - fradrag) + fradrag / 2;
        int diameter = startDiameter;
        int radius = diameter / 2;
        tegneflate.drawLine(forrigeX + radius, forrigeY + radius, x + radius, y + radius);
        antSirkler++;
        int denneX = x;
        int denneY = y;
        for (int sirkelnr = 0; sirkelnr < antSirkler; sirkelnr++) {
          tegneflate.drawOval(denneX, denneY, diameter, diameter);
          diameter += diameterInkr;
          denneX -= diameterInkr / 2;
          denneY -= diameterInkr / 2;
        }
        teller++;
      }
    }
  }
```

Kapittel 6.8

Oppgave 1

```
  public static int lesPositivtHeltall(String ledetekst) {
    int tall = lesHeltall(ledetekst);
    while (tall <= 0) {
      showMessageDialog(null, "Må ha et postivt heltall.");
      tall = lesHeltall(ledetekst);
    }
    return tall;
  }
```

I klientprogrammet:

```
  int posTall = DataLeser.lesPositivtHeltall("Skriv et positivt heltall: ");
  System.out.println(posTall);
```

Oppgave 2

```
  import java.util.Random;
  import static javax.swing.JOptionPane.*;
  class Oppg6_8_2 {
    public static void main(String[] args) {
      Random tallGen = new Random();

      /* Leser ytterste intervall */
      int grense1 = DataLeser.lesPositivtHeltall("Oppgi nederste grense: ");
      int grense4 = DataLeser.lesPositivtHeltall("Oppgi øverste grense: ");
      while (grense4 < grense1 + 2) {
        showMessageDialog(null,
                  "Øverste grense må være minst to større enn nederste.");
        grense1 = DataLeser.lesPositivtHeltall("Oppgi nederste grense: ");
        grense4 = DataLeser.lesPositivtHeltall("Oppgi øverste grense: ");
```

```
    }

    /* Leser innerste intervall */
    int grense2 =
            DataLeser.lesPositivtHeltall("Oppgi nederste grense i indre intervall: ");
    int grense3 =
            DataLeser.lesPositivtHeltall("Oppgi øverste grense i indre intervall: ");
    while (! (grense1 < grense2 && grense3 < grense4 && grense2 < grense3)) {
          showMessageDialog(null,
              "Det lille intervallet må ligge inne i det store [" + grense1 + ", " +
                 grense4 +
               "]. Dessuten må nedre grense være mindre enn øvre grense.");
    grense2 =
          DataLeser.lesPositivtHeltall("Oppgi nederste grense i indre intervall: ");
     grense3 = DataLeser.lesPositivtHeltall("Oppgi øverste grense i indre intervall: ");
    }

    int antOKTall = 0;
    int tall = grense1 + tallGen.nextInt(grense4 - grense1 + 1);

    while (tall < grense2 || tall > grense3) {
      antOKTall++;
      System.out.println(tall);
      tall = grense1 + tallGen.nextInt(grense4 - grense1 + 1);
    }
    System.out.println("Vi måtte trekke " + antOKTall +
                                " tall til vi fant et ugyldig: " + tall);
  }
}
```

Kapittel 6.10

Oppgave 1

```
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 2

```
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 6.11

Oppgave a)

```
    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.");
    }
```

Oppgave b)

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

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 */
    int svar = showConfirmDialog(null, "Er det skuddår?",
                                       "År", YES_NO_OPTION);
    if (svar == YES_OPTION) antDager[1] = 29;
    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]); // ingen linjeskift her
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

Unntak som kan kastes, og måter vi kan unngå det på:

- IndexOutOfBoundsException: Dette unntaket kastes hvis kopieringen medfører ugyldig indeksering i en av tabellene. Følgende test sørger for at dette unntaket ikke kastes:

```
if (fraIndeks >= 0 && tilIndeks >= 0 && antall >= 0 &&
   fraIndeks + antall <= fraTabell.length &&
   tilIndeks + antall <= tilTabell.length) {  // ok å kopiere
```

- ArrayStoreException: Dette unntaket kastes hvis datatypen til de to tabellene ikke stemmer overens, eller referansene ikke refererer til tabell-objekter. Eksempel:

```
int[] tab2 = {7, 14, -6, 0};
double[] tab3 = new double[10];
System.arraycopy(tab2, 1, tab3, 0, 3);  // kaster ArrayStoreException
```

  Tabellene må tilhøre samme primitive datatype.

- NullPointerException: Dette unntaket kastes hvis fraTabell eller tilTabell er lik null. Eksempel:

```
int[] tab1 = {1, 4, 6, -2};
int[] tab4 = null;
System.arraycopy(tab4, 1, tab1, 0, 3);  // kaster NullPointerException
```

Oppgave 3

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() {
  if (nedbør.length > 1) {
    double gjSnitt = finnGjSnitt();
    double sumKvadrat = 0;
    for (int i = 0; i < nedbør.length; i++) {
      sumKvadrat += (gjSnitt - nedbør[i]) * (gjSnitt - nedbør[i]);
    }
    System.out.println(sumKvadrat);
    double radikand = sumKvadrat/ (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);
```

Kapittel 7.4

Oppgave 1

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

```
import mittBibliotek.*;
class Oppg7_4_2 {
  public static int[] sorterOgFjernDubletter(int[] tab) {
    Sorter.sorterHeltallstabell(tab);  // sorterer først tabellen
    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 tabellslutt, 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[] 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
  } else return null; // retur, tabellene ikke like lange
}
```

Kapittel 7.5

Oppgave 1

```
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 2

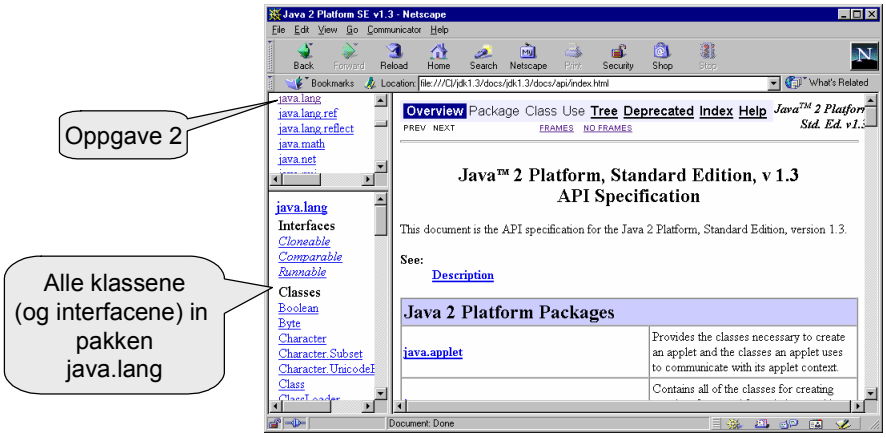
```
public static int søk(int[] tab, int verdi) {
  int indeks = 0;
  int indeksSøktEtter = -1;  // hvis vi ikke finner det vi leter etter
  while (indeks < tab.length && tab[indeks] <= verdi) {
    if (tab[indeks] == verdi) indeksSøktEtter = indeks;
    indeks++;
  }
  return indeksSøktEtter;
}
```

Kapittel 7.7

Oppgave 1

```
class Oppg7_7_1 {
  public static void main(String[] args) {
    int[] test = new int[8];
    java.util.Arrays.fill(test, 15);
    System.out.println("Har fylt tabellen med tallet 15:");
    for (int i = 0; i < test.length; i++) System.out.print(test[i] + " ");
    System.out.println();

    int fraIndeks = 3;
    int tilIndeks = 6;
    if (fraIndeks >= 0 && fraIndeks < test.length &&
        tilIndeks >= 0 && tilIndeks < test.length &&
        fraIndeks < tilIndeks) {
      java.util.Arrays.fill(test, fraIndeks, tilIndeks, 20);
      System.out.println("Har fylt tabellen med tallet 20 fra og med indeks "
                  + fraIndeks + " til og med indeks " + (tilIndeks - 1));
      for (int i = 0; i < test.length; i++) System.out.print(test[i] + " ");
      System.out.println();
    } else System.out.println("Ugyldig indeks.");
  }
}
/* Utskrift:
Har fylt tabellen med tallet 15:
15 15 15 15 15 15 15 15
Har fylt tabellen med tallet 20 fra og med indeks 3 til og med indeks 5
15 15 15 20 20 20 15 15
*/
```

Oppgave 2



Se til venstre på figuren. Du klikker på den aktuelle pakken øverst til venstre. Her har vi klikket på java.lang. Da kommer alle klassene i denne pakken fram.

Oppgave 3

Hovedforskjellen mellom String og StringBuilder er at mens String er immutabel, så er StringBuilder mutabel. Alle tilsynelatende endringsmetoder i klassen String endrer ikke på objektet, men lager et nytt. Eksempel:

```
String t1 = new String("Arne");
String t2 = t1.toUpperCase();
System.out.println(t1 + " " + t2);
```

Utskrift:

```
Arne ARNE
```

toUpperCase() lager et nytt objekt. For at vi skal kunne bruke det videre i programmet må vi lage en referanse (t2) som tar i mot referansen til dette objektet.

Klassen StringBuilder inneholder metoder for å fjerne og sette inn tegn i en tekst:

```
StringBuilder s1 = new StringBuilder("Arne");
StringBuilder s2 = s1.insert(2, "xxxx");
System.out.println(s1 + " " + s2);
```

Utskriften av denne lille kodebiten er som følger:

```
Arxxxxne Arxxxxne
```

Oppgave 4

Kompilerer følgende program:

```
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\Oppg7_7_4.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\Oppg7_7_4.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 7.8

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 Oppg7_8_1 {
  public static void main(String[] args) {
    Salgstall året2004 = new Salgstall("sko", 52, 5);  // oppgave a)
    året2004.settSalg(9, 0, 10000);  // oppgave b)
    året2004.settSalg(7, 3, 12100);  // oppgave c)
    System.out.println("Total salg uke 5: " + året2004.finnSalgForHelUke(4)); // d)
    System.out.println("Salg uke 6, mandag: " + året2004.finnSalg(5, 0));  // e)
    System.out.println("Totalt salg hele året: " + året2004.finnTotalsalg());  // f)
    int dag = året2004.finnMestLønnsommeUkedag();  // oppgave g)
    System.out.println("Mest lønnsomme ukedag: Dag nr. " + (dag + 1)
                  + " med salg på: " + året2004.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();
    else return 0;
```

```
}
```

Oppgave 2b

```
public int[] finnLønnsommeUker() {
  if (finnAntUker() > 0) {
    /* finner først maks.verdien */
    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;
  }
  else return null;
}
```

Oppgave 3a

Tabellen resultat:

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

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

Oppgave 3b

Fullstendig klasse med konstruktør:

```
class Oppg7_8_3 {
  private final int antMuligeRes = 3;
  private int antPoengVunnet = 3;
  private int antPoengUavgjort = 1;
  private int antPoengTap = 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 Oppg7_8_3(int antlag) {
  resultat = new int[antlag][antMuligeRes];
  /*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] * antPoengVunnet + resultat[i][1] * antPoengUavgjort +
                                          resultat[i][2] * antPoengTap;
  }
  return poengtabell;
}
}
```

Et testprogram kan se slik ut:

```
public static void main(String[] args) {
  Oppg7_8_3 fotball = new Oppg7_8_3(10);
  int[] poeng = fotball.finnAntPoeng();
  for (int i = 0; i < poeng.length; i++) {
    System.out.println("Lag nr. " + i + ": " + poeng[i] + " poeng");
  }
```

Kapittel 7.9

I klassen SalgBGS:

```
public void finnSalgUkedag() {
  int dag = DataLeser.lesHeltall(
              "Oppgi dagnr (min 1, maks " + salg.finnAntDgPrUke() + ")");
  int salget = salg.finnSalgForUkedag(dag);
  String melding = "";
  if (salget >= 0) melding += "Salget på ukedag nr " + dag + " er kr " + salget;
  else melding = "Ugyldig ukedag";
  showMessageDialog(null, melding);
}

public void finnMestLønnsommeUkedag() {
  int dag = salg.finnMestLønnsommeUkedag();
  showMessageDialog(null, "Mest lønnsomme ukedag er dag nr " + (dag + 1));
}
```

Og klassen som inneholder main() ser slik ut:

```
public static void main(String[] args) {
  Salgstall salg = new Salgstall("mat", 4, 5);
  SalgBGS bgs = new SalgBGS(salg);

  String[] alternativer = Menyvalg.lagAltTab();
  Object obj = showInputDialog(null, "Gjør et valg", "Salgstall",
              DEFAULT_OPTION, null, alternativer, alternativer[0]);
  Menyvalg valget = Menyvalg.finnValg((String) obj);
  while (valget != Menyvalg.valgAvslutt) {
    switch (valget) {
      case valgRegSalg:
        bgs.regSalg();
        break;

      case valgFinnSalg:
        bgs.finnSalg();
        break;

      case valgFinnSalgUke:
        bgs.finnSalgUke();
        break;

      case valgFinnTotalsalg:
        bgs.finnTotalsalg();
        break;

      case valgFinnSalgUkedag:
        bgs.finnSalgUkedag();
        break;

      case valgFinnLønnsomUkedag:
        bgs.finnMestLønnsommeUkedag();
        break;

      default:
        break;
    }
    obj = showInputDialog(null, "Gjør et valg", "Salgstall",
          DEFAULT_OPTION, null, alternativer, alternativer[0]);
    valget = Menyvalg.finnValg((String) obj);
  }
}
```

Vi måtte utvide enum-klassen Menyvalg med en klassemetode:

```
public static Menyvalg finnValg(String teksten) {
  Menyvalg[] altTab = Menyvalg.values();
  for (int i = 0; i < altTab.length; i++) {
    if ((altTab[i].tekst).equals(teksten)) return altTab[i];
  }
  return null; // skal ikke komme hit
```

```
}
```

Kapittel 7.10

Oppgave a)

```
int[][][] epler = new int[3][10][5];
```

Oppgave b)

Klassen har følgende variabler:

```
public static final int antallFelt = 3;
public static final int antallTrær = 10;
public static final int antallÅr = 5;
private int[][][] epler = new int[antallFelt][antallTrær][antallÅr];
```

Metoden som finner antall kilo pr. felt blir slik:

```
public int[] finnAvlingFordeltPåFelt() {
  int[] avling = new int[antallFelt];
  for (int felt = 0; felt < antallFelt; felt++) {
    for (int tre = 0; tre < antallTrær; tre++) {
      for (int år = 0; år < antallÅr; år++) {
        avling[felt] += epler[felt][tre][år];
      }
    }
  }
  return avling;
}
```

Kapittel 7.11

Oppgave 1

```
public int finnMaksimum() {
    int maks = 0;  // Husk at lokale variabler ikke automatisk får en startverdi.
    if (nedbør.length > 0) {
      maks = nedbør[0];
      /* I for-løkken nedenfor får vi med elementet med indeks 0,
         noe som egentlig er unødvendig (se original-metoden) */
      for (int dagNedbør : nedbør) {
        if (dagNedbør > maks) maks = dagNedbør;
      }
    }
    return maks;
  }
public double finnGjSnitt() {
    int sum = 0;
    for (int nedbørDag : nedbør) {
      sum += nedbørDag;
    }
   if (nedbør.length > 0) return (double) sum / (double) nedbør.length;
    else return 0.0;
  }
public int finnAntTørreDager() {
    int antall = 0;
```

```
    for (int nedbørDag : nedbør) {
      if (nedbørDag == 0) antall++;
    }
    return antall;
  }
```

Oppgave 2

Kun én metode i tillegg til finnTotalsalg() som er vist i boka side 273.

```
public int finnSalgForHelUke(int ukenr) {
  if (gyldigUkenr(ukenr)) {
    int sum = 0;
    int[] enUke = salg[ukenr];
    for (int beløp : enUke) {
      sum += beløp;
    }
    return sum;
  } else return -1;
}
```

Merk at vi ikke kan skrive om metoden finnSalgForUkedag(), på grunn av at den summerer kolonnevis.

Kapittel 7.12

Oppgave 1

Kodebiten vil 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].settPris(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

Med vanlig for-løkke:

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

```
}
```

Løkken skrevet om som utvidet for-løkke:

```
int sumLengde = 0;
for (String navn : navneliste) {
  sumLengde += navn.length();
}
System.out.println(sumLengde);
```

Oppgave 2d

Med vanlig for-løkke:

```
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);
```

Løkken skrevet om som utvidet for-løkke:

```
for (String navn : navneliste) {
  int indeks = navn.indexOf(tegn);
  while (indeks >= 0) {
    antTegn++;
    indeks = navn.indexOf(tegn, indeks + 1);
  }
}
```

Oppgave 3

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

Oppgave 4

```
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 (String etOrd : ord) {
    int lengde = etOrd.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 7.13

Løsningene til oppgave 1 og bruker ikke utvidet for-løkke, men se oppgave 3.

Oppgave 1a

```
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 1b

```
for (int i = 0; i < varer.length; i++) {
  System.out.println(varer[i].finnNavn());
 }
```

Oppgave 1c

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

Oppgave 1d

```
for (int i = 0; i < varer.length; i++) {
  double pris = varer[i].finnPris();
  pris *= 1.07;
  varer[i].settPris(pris);
}
```

Oppgave 2a

```
public int finnTotAntStudiepoeng() {
  int sum = 0;
  for (int i = 0; i < antFag; i++) {
    sum += fagene[i].finnAntStudiepoeng();
  }
  return sum;
}
```

Utprøving, se etter oppgave 2c)

Oppgave 2b

```
/* Metoden returnerer null dersom fag med denne koden ikke finnes */
public Fag finnFag(String kode) { // søk i tabell, se kap. 7.5
  for (int i = 0; i < antFag; i++) {
    String denneKoden = fagene[i].finnFagkode(); // fagkoden for fag med indeks i
    /* Sammenligner tekster, husk å bruke equals() (eller equalsIgnoreCase()) */
    if (denneKoden.equals(kode)) return fagene[i];
  }
  return null; // hit kommer vi dersom hele tabellen er gjennomsøkt uten resultat
}
```

Oppgave 2c

Del 1, utvidelse av tabellen:

```
public void registrerNyttFag1(String startFagkode,
            String startFagnavn, int startAntStudiepoeng) {
  if (antFag == fagene.length) { // full tabell, lager en ny og større
    Fag[] nyTab = new Fag[fagene.length + 2]; // liten økning for testformål
    System.out.println(  // bare for testformål
        "Utvider tabellen fra lengde " + fagene.length + " til lengde " + nyTab.length);
   for (int i = 0; i < antFag; i++) nyTab[i] = fagene[i]; // kopierer over eksisterende data
    fagene = nyTab;  // setter riktig referanse til å peke til den nye og større tabellen
  }
  /* legger inn data */
  fagene[antFag] = new Fag(startFagkode, startFagnavn, startAntStudiepoeng);
  antFag++;
}
```

Del 2, test på om faget er registrert fra før:

```
/* Metoden returnerer false dersom fag med denne koden er registrert fra før */
public boolean registrerNyttFag2(String startFagkode,
            String startFagnavn, int startAntStudiepoeng) {

  if (finnFag(startFagkode) != null) {  // bruker metoden fra oppgave d)
    return false; // RETUR, faget finnes fra før
  }
  registrerNyttFag1(startFagkode, startFagnavn, startAntStudiepoeng);
  return true;
}
```

Denne metoden heter registrerNyttFag2(). Den bruker metoden registrerNyttFag1() fra 1.del av oppgaven. I praksis ville man gjort registrerNyttFag1() privat.

Testprogram for alle metodene foran:

```
class Oppg7_13_2 {
  public static void main(String[] args) {
    Fagkatalog katalogen = new Fagkatalog(2);  // liten for testformål
    /* Registrerer først fem forskjellige fag, utskrift 5 x "true",
       samt melding om at tabellen utvides to ganger */
    System.out.println(
            katalogen.registrerNyttFag2("LV172D", "Programmering i Java", 6));
```

```
    System.out.println(
            katalogen.registrerNyttFag2("LV372D", "Publisering på Internett", 6));
    System.out.println(
             katalogen.registrerNyttFag2("LO451D", "Systemadministrasjon", 6));
    System.out.println(
            katalogen.registrerNyttFag2("BO715D", "Prosjekt", 9));
    System.out.println(
            katalogen.registrerNyttFag2("PO701D", "Prosjektteknikk", 6));

    /* Finner totalt antall studiepoeng */
    System.out.println(
                "Totalt antall studiepoeng: " + katalogen.finnTotAntStudiepoeng());

    /* Søker etter fag, utskrift skal bli info om LV172D og PO701D, samt null */
    System.out.println("Søker etter fag LV172D: " + katalogen.finnFag("LV172D"));
    System.out.println("Søker etter fag PO701D: " + katalogen.finnFag("PO701D"));
    System.out.println("Søker etter fag LO173D: " + katalogen.finnFag("LO173D"));

    /* Registrerer to fag som allerede er registrert, skal få false */
    System.out.println(katalogen.registrerNyttFag2("LO451D", "Systemering", 12));
    System.out.println(katalogen.registrerNyttFag2("BO715D", "Prosjekt 2", 12));

    /* Utskrift av alle registrert fag */
    String utskrift = "";
    for (int i = 0; i < katalogen.finnAntallFag(); i++) {
      utskrift += katalogen.finnFag(i).toString() + "\n";
    }
    System.out.println(utskrift);
  }
}

/* Utskrift:
true
true
Utvider tabellen fra lengde 2 til lengde 4
true
true
Utvider tabellen fra lengde 4 til lengde 6
true
Totalt antall studiepoeng: 33
Søker etter fag LV172D: Kode: LV172D, Navn: Programmering i Java, 6
studiepoeng.
Søker etter fag PO701D: Kode: PO701D, Navn: Prosjektteknikk, 6 studiepoeng.
Søker etter fag LO173D: null
false
false
Kode: LV172D, Navn: Programmering i Java, 6 studiepoeng.
Kode: LV372D, Navn: Publisering på Internett, 6 studiepoeng.
Kode: LO451D, Navn: Systemadministrasjon, 6 studiepoeng.
Kode: BO715D, Navn: Prosjekt, 9 studiepoeng.
Kode: PO701D, Navn: Prosjektteknikk, 6 studiepoeng.
```

```
*/
```

Oppgave 3

1b) kan skrives om slik:

```
for (Vare enVare : varer) {
  System.out.println(enVare.finnNavn());
}
```

1c) egner seg ikke for utvidet for-løkke, da gjennomløpet starter på indeks 1, og vi bruker indeksen inne i løkken.

1d) Kan ikke bruke utvidet for-løkke her på grunn av at elementene i tabellen skal forandres.

Kan ikke bruke utvidet for-løkke i oppgave 2 på grunn av at bare de antFag første elementene i tabellen gjennomløpes.

Kapittel 8.3

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.

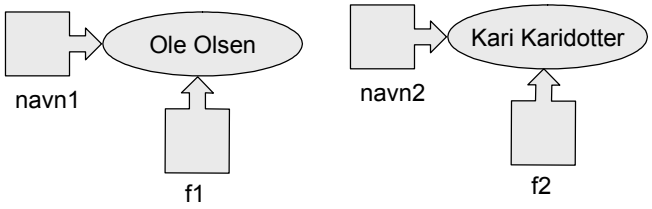
Oppgave 3

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 Oppg8_3_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
```

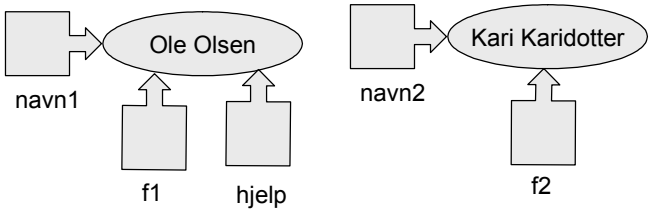
```
*/
```

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

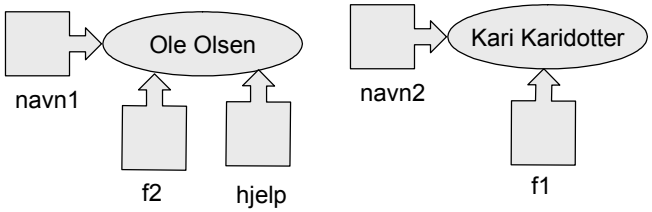


Hva skjer inne i metoden? Jo, vi oppretter en ny referanse, hjelp, som vi setter til å peke til det samme som f1 peker til:

```
Navn hjelp = f1;
```

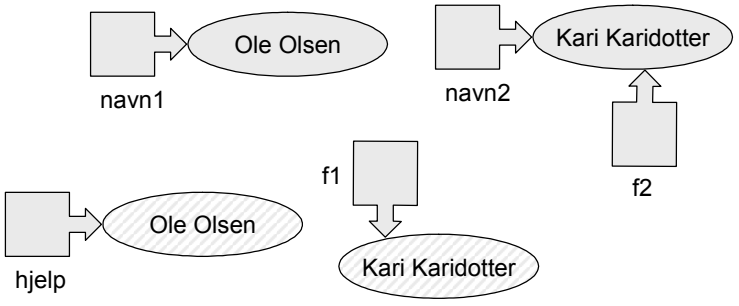


Vi bruker hjelp til å bytte om referansene slik at sluttresultatet blir som følger:



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

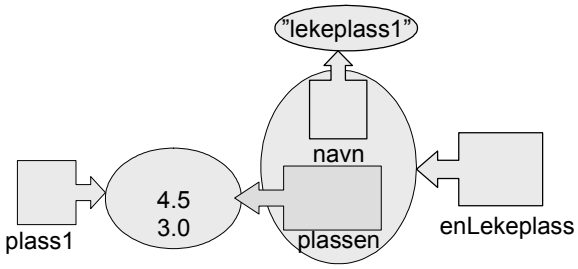
Dersom vi skal endre på objekter som tilhører klienten, må vi *ikke* flytte på referansene inne i metoden:

```
public void byttOmObjekter3(Navn f1, Navn f2) {
  Navn temp = new Navn(f1.finnFornavn(), f1.finnEtternavn());
  f1.settFornavn(f2.finnFornavn());
  f1.settEtternavn(f2.finnEtternavn());
  f2.settFornavn(temp.finnFornavn());
  f2.settEtternavn(temp.finnEtternavn());
}
```

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

Kapittel 8.4

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 8.5

Eksempel på bruk av metodene, se etter oppgave 4.

Oppgave 1

```
public boolean likeGammel(Person p) {
  if (p.fødselsår == fødselsår) return true;  // kan også bruke p.finnFødselsår()
  else return false;
}
```

Oppgave 2

```
public int sammenliknAlder(Person p) {
```

```
    if (fødselsår < p.fødselsår) return -1;
    else if (fødselsår > p.fødselsår) return +1;
    else return 0;
  }
```

Oppgave 3

```
  public boolean flereÅrEldreEnn(Person p, int antÅr) {
    if (fødselsår - antÅr > p.fødselsår) return true;
    else return false;
  }
```

Oppgave 4

```
  public boolean equals(Object obj) {
    if (!(obj instanceof Person)) return false;
    if (this == obj) return true;

    Person p = (Person) obj;
    if (navn.equals(p.finnNavn())) return true;
    else return false;
  }

  public int compareTo(Person p) {
    return (navn.compareTo(p.finnNavn()));
  }
```

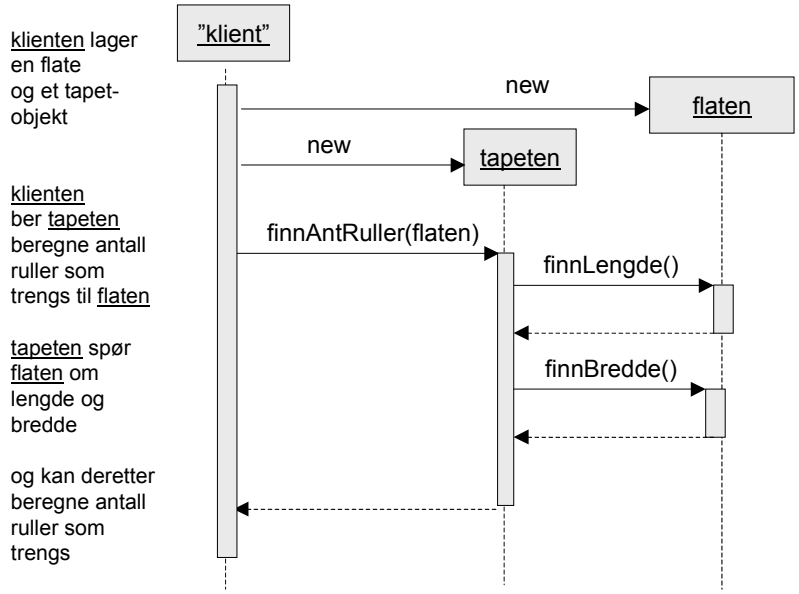
Eksempel på bruk av metodene:

```
  class Oppg8_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.");

      System.out.println(p1.compareTo(p2));  // positivt tall
      System.out.println(p1.compareTo(p1));  // 0
      System.out.println(p2.compareTo(p1));  // negativt tall
    }
  }

  /* Utskrift:
  Ikke like gamle
  p1 er yngst
  Ikke likhet.
```

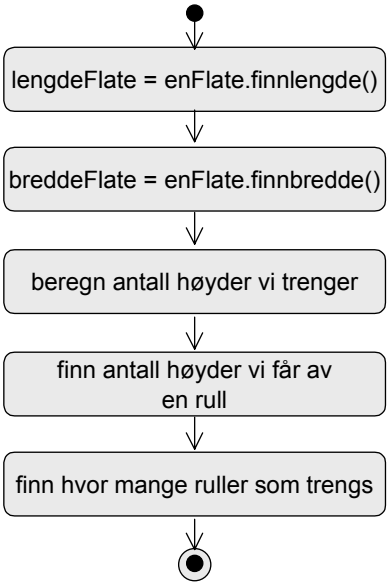
```
Likhet
4
0
-4
*/
```

Kapittel 8.6

Oppgave 1a



Oppgave 1b



Oppgave 2

```
/*
 * Maling.java   E.L. 2004-04-03
 *
 * Klassen Maling tilbyr metoder for beregning av materialbehov og pris.
 */
class Maling {
  private static final double grense = 0.02;  // grense for å kjøpe 0.5 liter maling til
  private String navn;  // identifiserer malingen
  private double pris;  // pr.liter
  private int antStrøk;
  private double antKvmPrLiter;

  public Maling(String startNavn, double startPris, int startAntStrøk,
                double startAntKvmPrLiter) {
    navn = startNavn;
    antStrøk = startAntStrøk;
    antKvmPrLiter = startAntKvmPrLiter;
    pris = startPris;
  }

  public String finnNavn() {
    return navn;
  }

  public double finnPrisPrLiter() {
    return pris;
  }

  public int finnAntStrøk() {
    return antStrøk;
  }

  public double finnAntKvmPrLiter() {
    return antKvmPrLiter;
  }

  /*
   * Metoden finner antall liter maling som trengs,
   * behovet rundes oppover til nærmeste halve liter.
   */
  public double finnAntLiter(Flate enFlate) {
    double areal = enFlate.finnAreal();
    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;
  }

  public double finnTotalpris(Flate enFlate) {
```

```
    return finnAntLiter(enFlate) * pris;
  }

  public String toString() {
    java.text.DecimalFormat utskriftFormat =
                              new java.text.DecimalFormat("####0.00");
    return "Maling: " + navn + ", pris kr " + utskriftFormat.format(pris) +
          " pr. liter, ant. strøk: " + antStrøk + ", ant. kvm/l: "  +
          utskriftFormat.format(antKvmPrLiter);
  }
}
```

Kapittel 9.1

```
import java.util.*;
import mittBibliotek.DataLeser; // kap. 6.8
import static javax.swing.JOptionPane.*;
class Oppg9_1_1til4 {
  public static void main(String[] args) {

    /* 1 */
    ArrayList<Vare> varer = new ArrayList<Vare>();
    int varenr = 100;
    String varenavn = DataLeser.lesTekst("Oppgi varenavn (avslutt med \"X\"): ");
    varenavn = varenavn.trim();
    while (!varenavn.equalsIgnoreCase("X")) {
      double pris = DataLeser.lesDesimaltall("Oppgi pris uten moms: ");
      Vare enVare = new Vare(varenavn, varenr, pris);
      varer.add(enVare);
      varenr++;
      varenavn = DataLeser.lesTekst("Oppgi varenavn (avslutt med \"X\"): ");
      varenavn = varenavn.trim();
    }

    /* 2, vanlig for-løkke */
    String resultat = "";
    for (int i = 0; i < varer.size(); i++) {
      Vare enVare = varer.get(i);
      resultat += (enVare.finnNavn() + ", nr.: " + enVare.finnVarenr()
                   + ", pris kr.: " + enVare.finnPris() + "\n");
    }
    showMessageDialog(null, resultat);

    /* 2, utvidet for-løkke */
    resultat = "";
    for (Vare enVare : varer) {
      resultat += (enVare.finnNavn() + ", nr.: " + enVare.finnVarenr()
                   + ", pris kr.: " + enVare.finnPris() + "\n");
    }
    showMessageDialog(null, resultat);
```

```
    /* 3 */
    String søkenavn = DataLeser.lesTekst("Oppgi navn på vare det skal søkes etter:
");
    søkenavn = søkenavn.trim();
    boolean funnet = false;
    Vare varen = null;
    int indeks = 0;
    while (indeks < varer.size() && !funnet) {
      varen = varer.get(indeks);
      String navn = varen.finnNavn();
      if (navn.equals(søkenavn)) funnet = true;
      else {
        /*
         * Obs! oppdaterer indeks bare dersom varen ikke finnes,
         *  vi bruker indeksen i oppgave 4
         */
        indeks++;
      }
    }
    if (funnet) {
      showMessageDialog(null, "Ytterligere opplysninger om denne varen: nr.: "
                    + varen.finnVarenr() + ", pris kr.: " + varen.finnPris());
    } else {
      showMessageDialog(null,
          "Varen med navn " + søkenavn + " finnes ikke i registeret.");
    }

    /* 4 */
    if (funnet) varer.remove(indeks);

    /* Skriver ut alle varene */
    resultat = "";
    for (Vare enVare : varer) {
      resultat += (enVare.finnNavn() + ", nr.: " + enVare.finnVarenr()
                    + ", pris kr.: " + enVare.finnPris() + "\n");
    }
    showMessageDialog(null, resultat);
  }
}
```

Kapittel 9.3

Oppgave 1

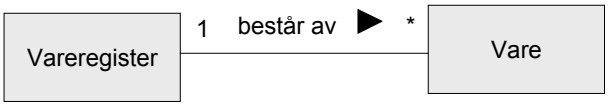
```
Integer etHeltall = new Integer(15); // oppgave a)
Character forbokstav = new Character('S'); // oppgave b)
int tall = etHeltall.intValue(); // oppgave c)
String tekst1 = forbokstav.toString(); // oppgave d)
String tekst2 = etHeltall.toString(); // oppgave e)
```

Oppgave 2

```
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 9.4

Oppgave 1

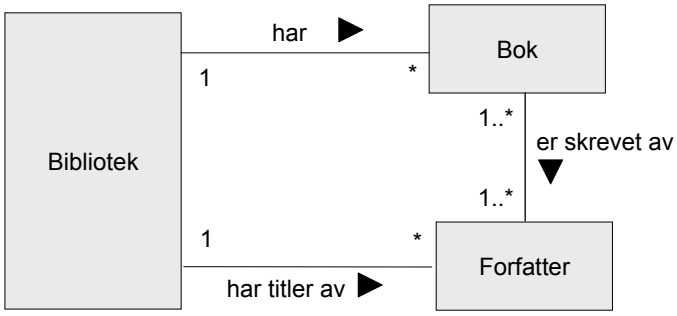


Kan eventuelt tegne aggregering.

Operasjoner knyttet til klassen Vareregister kan for eksempel være:

- Registrer ny vare.
- Finn prisen på en gitt vare.
- Lag en oversikt over alle varene.
- Fjern en vare fra registeret.
- Endre prisen på en vare.

Oppgave 2



Kapittel 9.5

Oppgave 1

```
public boolean fjernFlate(String flatenavn) {
  for (int i = 0; i < alleFlater.size(); i++) {
    Flate denne = alleFlater.get(i);
    if ((denne.finnNavn()).equals(flatenavn)) {
      alleFlater.remove(i);
      return true; // RETUR, flate fjernet
    }
  }
  return false; // ingen flate med dette navnet er funnet
```

```
}
```

Oppgave 2

```
public boolean fjernMaling(String malingnavn) {
 for (int malingsIndeks = 0; malingsIndeks < alleMalingstyper.size();
                                                          malingsIndeks++) {
   Maling denneMaling = alleMalingstyper.get(malingsIndeks);
   if ((denneMaling.finnNavn()).equals(malingnavn)) {

    /* Brukes malingen? */
    for (Flate denneFlate : alleFlater) {
      if (denneFlate.finnMaling() == denneMaling) {
        return false;  // RETUR. Malingen er i bruk.
      }
    }
    alleMalingstyper.remove(malingsIndeks);
    return true; // RETUR. Maling fjernet.
   }
 }
 return false; // RETUR. Ingen maling med dette navnet.
}
```

Testprogrammet kan se slik ut:

```
public static void main(String[] args) {
  Oppussingsprosjekt prosjekt = new Oppussingsprosjekt("Oppgave");
  Maling m1 = new Maling("Heimdal Ekstra", 3, 10, 100);
  Maling m2 = new Maling("Heimdal Super", 2, 12, 80);
  Flate f1 = new Flate("tak", 6, 1);
  Flate f2 = new Flate("vegg", 4, 3);
  f1.settMaling(m1);
  f2.settMaling(m2);
  prosjekt.registrerNyFlate(f1);
  prosjekt.registrerNyFlate(f2);
  prosjekt.registrerNyMaling(m1);
  prosjekt.registrerNyMaling(m2);

  if (!prosjekt.fjernFlate("golv")) System.out.println("golv ikke fjernet");
  if (!prosjekt.fjernMaling("HHH")) System.out.println("HHH ikke fjernet");
  if (!prosjekt.fjernMaling("Heimdal Super")) {
    System.out.println("Heimdal Super ikke fjernet");
  }

  /* skal skifte maling på "vegg" */
  f2.settMaling(m1);

  /* Nå skal m2 kunne slettes */
  if (prosjekt.fjernMaling("Heimdal Super")) {
     System.out.println("Heimdal Super er fjernet");
  }
  if (prosjekt.fjernFlate("tak")) System.out.println("tak fjernet");

  /* Kontrollutskrifter */
```

```
    for (int i = 0; i < prosjekt.finnAntFlater(); i++) {
      Flate flate = prosjekt.finnFlate(i);
      System.out.println("Flate med indeks: " + i + ": " + flate.finnNavn());
    }

    for (int i = 0; i < prosjekt.finnAntMalinger(); i++) {
      Maling m = prosjekt.finnMaling(i);
      System.out.println("Maling med indeks: " + i + ": " + m.finnNavn());
    }
  }
```

Utskriften ser slik ut:

```
golv ikke fjernet
HHH ikke fjernet
Heimdal Super ikke fjernet
Heimdal Super er fjernet
tak fjernet
Flate med indeks: 0: vegg
Maling med indeks: 0: Heimdal Ekstra
```

Kapittel 9.7

Oppgave 1

Endringer i klassehodet til klassen Vare:

```
class Vare implements Comparable<Vare> {
```

og en ny metode i klassen:

```
public int compareTo(Vare annenVare) {
  if (varenr < annenVare.varenr) return -1;
  else if (varenr > annenVare.varenr) return 1;
  else return 0;
}
```

Testprogram:

```
  public static void main(String[] args) {
    Vare v1 = new Vare("ost", 100, 70);
    Vare v2 = new Vare("pølse", 101, 60);
    Vare v3 = new Vare("banan", 101, 9.50);

    /* Skal gi utskriften -1 1 0 */
    System.out.println(v1.compareTo(v2));
    System.out.println(v2.compareTo(v1));
    System.out.println(v2.compareTo(v2));
  }
```

Oppgave 2a

```
String[] navnetabell = {"Æsop", "Einar", "Åsmund", "Petter", "Øyvind", "Synnøve"};
```

Oppgave 2b

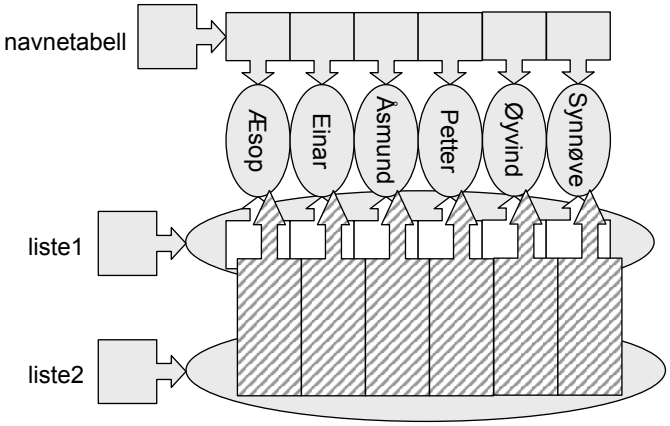
```
ArrayList<String> liste1 = new ArrayList<String>();
for (String navn : navnetabell) {
  liste1.add(navn);
```

```
}
```

Oppgave 2c

```
ArrayList<String> liste2 = new ArrayList<String>();
for (String navn : liste1) {
  liste2.add(navn);
}
```

Oppgave 2d



Det er ikke enkelt å tegne dette, men vi har altså tre referanser til hvert objekt. Fra henholdsvis tabellen navnetabell, og fra tabell-listene liste1 og liste2.

Oppgave 2e

Sortering uten kollator:

```
Collections.sort(liste2);  // uten kollator
```

Resultat: Einar, Petter, Synnøve, Åsmund, Æsop, Øyvind

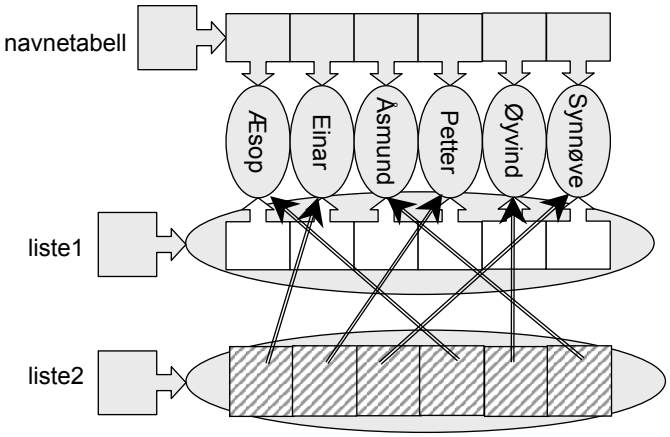
Sortering med kollator:

```
Collator kollator = Collator.getInstance();
Collections.sort(liste2, kollator);
```

Resultat: Einar, Petter, Synnøve, Æsop, Øyvind, Åsmund

Vi ser at vi må bruke kollator for at sorteringen skal bli riktig.

Oppgave 1f



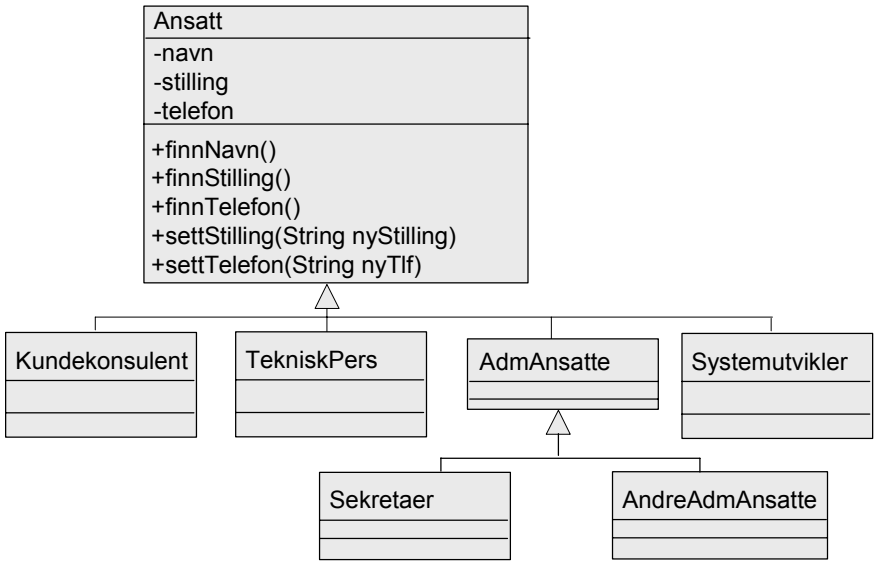
Figuren viser at referansene i den sorterte tabell-listen, liste2, nå peker til objektene i sortert rekkefølge.

Oppgave 3

```
public static void main(String[] args) {
  Vare[] varer = new Vare[5];
  varer[0] = new Vare("ost", 200, 70);
  varer[1] = new Vare("pølse", 111, 60);
  varer[2] = new Vare("banan", 101, 9.50);
  varer[3] = new Vare("tomat", 121, 35);
  varer[4] = new Vare("syltetøy", 115, 50);

  java.util.Arrays.sort(varer); // forutsetter at Vare implementerer Comaparable

  for (Vare enVare : varer) {
    System.out.println("Nr: " + enVare.finnVarenr() + ", navn: " +
                       enVare.finnNavn() +  ", pris: " + enVare.finnPris());
  }
}
```

Kapittel 10.1

Oppgave 1a



Vi har valgt å lage en klasse som heter AndreAdmAnsatte. Til denne klassen hører alle administrativt ansattte som ikke er sekretærer. Alternativt kunne vi sløyfet denne klassen, og i stedet latt disse objektene tilhøre klassen AdmAnsatte. Imidlertid ville dette kunne skapt forvirring, for klassen AdmAnsatte beskriver jo alle administrativt ansatte, også sekretærer. En god rettesnor for modellering er å la alle objekter tilhøre klasser på laveste nivå i klassetreet.

Oppgave 1b

Klassene Kundekonsulent, TekniskPers, AdmAnsatte, Sekretaer, AndreAdmAnsatte og Systemutvikler arver følgende metoder fra klassen Ansatt: finnNavn(), finnStilling(), finnTelefon(), settStilling(), settTelefon().

Kapittel 10.2

Oppgave 1

```
class Ekstern extends Person {
  private String funksjon;
  private double lønnHittil = 0.0;
  public static final java.text.NumberFormat pengeformat =
                java.text.NumberFormat.getCurrencyInstance();

  public Ekstern(long startFnr, String startNavn, String startAdr,
     String startFunksjon) {
```

```
    super(startFnr, startNavn, startAdr);
    funksjon = startFunksjon;
  }

  public String finnFunksjon() {
    return funksjon;
  }

  public double finnLønnHittil() {
    return lønnHittil;
  }

  public void endreLønnHittil(double endring) {
    lønnHittil += endring;
  }
  public String toString() {
    return "Eksternt tilknyttet person med funksjon: " + funksjon +
          ", lønn hittil: " + pengeformat.format(lønnHittil) + " " + super.toString();
  }
}

class Oppg10_2_1 {
  public static void main(String[] args) {
    Ekstern eksternPerson = new Ekstern(12106078756L, "Ole Pettersen",
      "Storgt 3, 7001 Trondheim", "timelærer");
    System.out.println(
       "Prøver finn-metoder: " +  eksternPerson.finnFnr() + " " +
        eksternPerson.finnNavn() + " " + eksternPerson.finnAdresse() + " " +
        eksternPerson.finnFunksjon() + " " +
         Ekstern.pengeformat.format(eksternPerson.finnLønnHittil()));
    eksternPerson.endreLønnHittil(2000);
    eksternPerson.endreLønnHittil(4000);
    System.out.println("Prøver toString(): " + eksternPerson);
  }
}

/* Utskrift:
Prøver finn-metoder: 12106078756 Ole Pettersen Storgt 3, 7001 Trondheim
timelærer kr 0,00
Prøver toString(): Eksternt tilknyttet person med funksjon: timelærer, lønn hittil:
kr 6.000,00 f.nr.: 12106078756, navn: Ole Pettersen, adresse: Storgt 3, 7001
Trondheim
*/
```

Oppgave 2

```
/*
 * Klassen Ansatt er mutabel, stilling og telefon kan endres.
 * Vi koder subklassene Kundekonsulent og AdmAnsatte med subklasser.
 * Vi koder ikke klassene Systemutvikler og TekniskPers,
 * de blir helt tilsvarende klassen Kundekonsulent.
 */
```

```
abstract class Ansatt {
  private String navn;
  private String stilling;
  private String telefon;
  public Ansatt(String startNavn, String startStilling, String startTelefon) {
    navn = startNavn;
    stilling = startStilling;
    telefon = startTelefon;
  }

  public String finnNavn() {
    return navn;
  }

  public String finnStilling() {
    return stilling;
  }

  public String finnTelefon() {
    return telefon;
  }

  public void settStilling(String nyStilling) {
    stilling = nyStilling;
  }

  public void settTelefon(String nyTelefon) {
    telefon = nyTelefon;
  }

  public void beregnLønn() {
    System.out.println("Lønn regnes ut i følge tabell");
  }

  public String toString() {
    return "Navn: " + navn + ", stilling: " + stilling + ", telefon: " + telefon;
  }
}

class Kundekonsulent extends Ansatt {
  public Kundekonsulent(String startNavn, String startStilling, String startTelefon) {
    super(startNavn, startStilling, startTelefon);
  }

  public String toString() {
    return "Kundekonsulent med følgende data: " + super.toString();
  }
}

class AdmAnsatte extends Ansatt {
  public AdmAnsatte(String startNavn, String startStilling, String startTelefon) {
```

```
    super(startNavn, startStilling, startTelefon);
  }
  /* Lager ikke toString()-metode her, utgaven arvet fra Ansatt gjelder derfor. */
}

class Sekretaer extends AdmAnsatte {
  public Sekretaer(String startNavn, String startStilling, String startTelefon) {
    super(startNavn, startStilling, startTelefon);
  }

  public String toString() {
    /*
     * I setningen nedenfor kaller vi den toString() som er arvet fra superklassen,
     * det vil si fra klassen AdmAnsatte. Nå er det ikke laget noen slik metode der,
     * det vil derfor være den toString() som tilhører Ansatt som blir kalt.
     */
    return "Sekretær med følgende data: " + super.toString();
  }
}

class AndreAdmAnsatte extends AdmAnsatte {
  public AndreAdmAnsatte(String startNavn, String startStilling,
                                                    String startTelefon) {
    super(startNavn, startStilling, startTelefon);
  }

  public String toString() {
   return
    "En administrativt ansatt som ikke er sekretær, med følgende data: " +
      super.toString();
  }
}
```

Oppgave 3

```
class Oppg10_2_3 {
  public static void main(String[] args) {
    Kundekonsulent konsulent =
        new Kundekonsulent("Ole Ås", "Trondheimskontoret", "56766");
    Sekretaer sekr = new Sekretaer("Anne Nilsen", "Stilling 123", "45345");
    AndreAdmAnsatte ikkeSekr =
         new AndreAdmAnsatte("Anne Nilsen", "Stilling 123", "45345");

    /* Prøver alle arvede metoder */
    konsulent.settStilling("Oslokontoret");
    konsulent.settTelefon("45454");
    System.out.println("Kundekonsulenten: " + konsulent.finnNavn() + ", "
      + konsulent.finnStilling() + ", " + konsulent.finnTelefon());

    sekr.settStilling("Stilling 999");
    sekr.settTelefon("67676");
    System.out.println("Sekretæren: " + sekr.finnNavn() + ", "
      + sekr.finnStilling() + ", " + sekr.finnTelefon());
```

```
    ikkeSekr.settStilling("Stilling 777");
    ikkeSekr.settTelefon("888888");
    System.out.println("Ikke-sekretæren: " + ikkeSekr.finnNavn() + ", "
      + ikkeSekr.finnStilling() + ", " + ikkeSekr.finnTelefon());

    /* Prøver metoden beregnLønn()*/
    konsulent.beregnLønn();
    sekr.beregnLønn();
    ikkeSekr.beregnLønn();

    /* Prøver toString() */
    System.out.println(konsulent);
    System.out.println(sekr);
    System.out.println(ikkeSekr);
  }
}

/* Utskrift:
Kundekonsulenten: Ole +s, Oslokontoret, 45454
Sekretµren: Anne Nilsen, Stilling 999, 67676
Ikke-sekretµren: Anne Nilsen, Stilling 777, 888888
Lønn regnes ut i følge tabell
Lønn regnes ut i følge tabell
Lønn regnes ut i følge tabell
Kundekonsulent med følgende data: Navn: Ole Ås, stilling: Oslokontoret, telefon:
 45454
Sekretær med følgende data: Navn: Anne Nilsen, stilling: Stilling 999, telefon:
67676
En administrativt ansatt som ikke er sekretær, med følgende data:
Navn: Anne Nilsen, stilling: Stilling 777, telefon: 888888
*/
```

Kapittel 10.3

Oppgave 1

I klassen Kundekonsulent:

```
public void beregnLønn() {
  System.out.println("Lønn for kundekonsulent er ennå ikke implementert");
}
```

I klassen AdmAnsatte:

```
public void beregnLønn() {
  System.out.println("Lønn for administrativt ansatte er ennå ikke implementert");
}
}
```

Denne metoden vil arves av klassene Sekretaer og AndreAdmAnsatte. Dersom vi senere får lagt inn en ordentlig lønnsberegning for, for eksempel sekretærer, vil den metoden bli brukt i stedet, og metoden foran vil arves kun av AndreAdmAnsatte.

Oppgave 2

```
class Oppg10_3_2 {
  public static void main(String[] args) {
    Ansatt[] ansatte = new Ansatt[5];
    ansatte[0] = new AndreAdmAnsatte("Hanne By", "Trondheim", "23456");
    ansatte[1] = new Kundekonsulent("Inger Ås", "Oslo 2", "98767");
    ansatte[2] = new Sekretaer("Åge Jensen", "NT", "67456");
    ansatte[3] = new AndreAdmAnsatte("Eva Hansen", "Novell", "89563");
    ansatte[4] = new Kundekonsulent("Torunn Ski", "Oslo 2", "78452");
    for (Ansatt enAnsatt : ansatte) {
      System.out.println(enAnsatt.finnNavn() + " lønn: ");
      enAnsatt.beregnLønn();
    }
  }
}

/* Utskrift:
Hanne By lønn:
Lønn for administrativt ansatte er ennå ikke implementert
Inger Ås lønn:
Lønn for kundekonsulent er ennå ikke implementert
Åge Jensen lønn:
Lønn for administrativt ansatte er ennå ikke implementert
Eva Hansen lønn:
Lønn for administrativt ansatte er ennå ikke implementert
Torunn Ski lønn:
Lønn for kundekonsulent er ennå ikke implementert
*/
```

Oppgave 3

```
class Oppg10_3_3 {
  public static void main(String[] args) {
    java.util.ArrayList<Ansatt> ansatte = new java.util.ArrayList<Ansatt>();
    ansatte.add(new AndreAdmAnsatte("Hanne By", "Trondheim", "23456"));
    ansatte.add(new Kundekonsulent("Inger Ås", "Oslo 2", "98767"));
    ansatte.add(new Sekretaer("Åge Jensen", "NT", "67456"));
    ansatte.add(new AndreAdmAnsatte("Eva Hansen", "Novell", "89563"));
    ansatte.add(new Kundekonsulent("Torunn Ski", "Oslo 2", "78452"));
    for (Ansatt enAnsatt : ansatte) {
      System.out.println(enAnsatt.finnNavn() + " lønn: ");
      enAnsatt.beregnLønn();
    }
  }
}
```

Oppgave 4

for-løkken fra oppgave 3 ser nå slik ut:

```
for (Ansatt enAnsatt : ansatte) {
  String kategori;
  if (enAnsatt instanceof Kundekonsulent) kategori = "Kundekonsulent";
  else if (enAnsatt instanceof Sekretaer) kategori = "Sekretær";
```

```
    else kategori = "Adm.ansatt, men ikke sekretær";
    System.out.println("Kategori: " + kategori + ", " + enAnsatt.finnNavn() + " lønn: ");
    enAnsatt.beregnLønn();
  }
```

Oppgave 5

I klassen Ansatt har vi en abstrakt metode:

```
public abstract String finnKlassenavn();
```

Implementasjon i klassen Kundekonsulent:

```
public String finnKlassenavn() {
  return "Kundekonsulent";
}
```

Implementasjon i klassen Sekretaer:

```
public String finnKlassenavn() {
  return "Sekretær";
}
```

Implementasjon i klassen AndreAdmAnsatte:

```
public String finnKlassenavn() {
  return "AndreAdmAnsatte";
}
```

Løkken fra oppgave 3 ser nå slik ut:

```
for (Ansatt enAnsatt : ansatte) {
  String kategori  = enAnsatt.finnKlassenavn();
  System.out.println("Kategori: " + kategori + ", " + enAnsatt.finnNavn() + " lønn: ");
  enAnsatt.beregnLønn();
}
```

Kapittel 10.4

Oppgave 1

Dersom metoden beregnLønn() er abstrakt i klassen Ansatt, medfører det at også klassen er abstrakt. Det vil si at det ikke er mulig å lage objekter av klassen. Metoden vil i tilfelle ikke ha noen implementasjon som vil kunne arves til subklasser. Hver subklasse (som det skal være mulig å lage objekter av) må ha en implementasjon av metoden.

Dersom metoden ikke er abstrakt, kan vi lage en standardutgave som blir brukt dersom en subklasse ikke har sin egen utgave av metoden. Det kan godt hende dette er ønskelig.

Vi kan likevel lage klassen abstrakt dersom vi vil forhindre at det lages objekter av klassen. Vi krever dermed at en klient må vite hvilken kategori ansatt en person tilhører før han/hun kan registreres.

Oppgave 2

Metoden er abstrakt i klassen Figur:

```
public abstract double finnOmkrets();
```

Implementasjon i klassen Kvadrat:

```
public double finnOmkrets() {
  return 4 * side;
}
```

Implementasjon i klassen Trekant:

```
public double finnOmkrets() {
  double hypotenus = Math.sqrt(grunnlinje * grunnlinje + høyde * høyde);
  return grunnlinje + høyde + hypotenus;
}
```

Implementasjon i klassen Sirkel:

```
public double finnOmkrets() {
  return 2 * Math.PI * radius;
}
```

Utvider kroppen i for-løkken i klientprogrammet:

```
System.out.print("Omkretsen er " + figurtabell[i].finnOmkrets() + ", ");
```

Kapittel 10.5

Oppgave 1

```
class BokstavSamling {
  private java.util.ArrayList<A>bokstaver = new java.util.ArrayList<A>();

  void registrerNyBokstav(A etBokstavObjekt) {
    bokstaver.add(etBokstavObjekt);
  }

  int finnAntallB() {
    int antall = 0;
    for (A obj : bokstaver) {
      if (obj instanceof B) antall++;
    }
    return antall;
  }
}

class Oppg10_5_1 {
  public static void main(String[] args) {
    BokstavSamling bokstaver = new BokstavSamling();
    /* Kan ikke lage objekter av klassene A og B, de er abstrakte */
    bokstaver.registrerNyBokstav(new C());
    bokstaver.registrerNyBokstav(new E());
    bokstaver.registrerNyBokstav(new D());
    bokstaver.registrerNyBokstav(new F());
    bokstaver.registrerNyBokstav(new D());
    bokstaver.registrerNyBokstav(new E());
    bokstaver.registrerNyBokstav(new E());
    System.out.println("Vi har " + bokstaver.finnAntallB() +
                                  " objekter som tilhører B eller subklasser av B");
  }
```

```
}

/* Utskrift:
Vi har 4 objekter som tilhører B eller subklasser av B
*/
```

Kapittel 10.6

Oppgave 2

Klassen TestKlasse3 har to medlemmer. Metoden metode1() er en erstatning for metoden med samme signatur arvet fra klassen TestKlasse1. Metoden metode3() er deklarert i klassen TestKlasse3.

Setningen obj1.metode3(); i main() gir kompileringsfeil. Årsaken er at obj1 tilhører klassen TestKlasse1, og denne klassen har ikke noe medlem med navn metode3().

Utskrift fra kjøring av programmet ser slik ut:

```
Metode1
Metode 1 i TestKlasse3
Metode 3 i TestKlasse3
Metode 1 i TestKlasse3
Metode 3 i TestKlasse3
```

Kapittel 10.8

Oppgave 1

```
class Oppg10_8_1 {
  public static void main(String[] args) {
    Flate golv = new Flate("Stuegolv", 5, 4);
    Materiale[] materialer = new Materiale[3];
    ForsteSortBelegg belegg1 = new ForsteSortBelegg("PrimaVare", 200, 5);
    AnnenSortBelegg belegg2A = new AnnenSortBelegg("SekundaVare1", 200, 5);
    AnnenSortBelegg belegg2B = new AnnenSortBelegg("SekundaVare2", 200, 6);
    materialer[0] = belegg1;
    materialer[1] = belegg2A;
    materialer[2] = belegg2B;
    for (Materiale etMateriale : materialer) {
      System.out.print("Belegg: " + etMateriale.finnNavn());
      System.out.println(", materialbehov: "
              + etMateriale.finnMaterialbehov(golv) + " m, pris kr.: " +
                 etMateriale.finnTotalpris(golv));
    }
  }
}
```

Oppgave 2

Metoden toString() i klassen ForsteSortBelegg:

```
public String toString() {
  return super.toString() + ", dette er 1.sort belegg.";
}
```

Metoden toString() i klassen AnnenSortBelegg:

```
public String toString() {
  return super.toString() + ", dette er 2.sort belegg.";
}
```

Løkken i klientprogrammet i oppgave 1 utvides med følgende setning:

```
System.out.println(etMateriale);  // toString() er underforstått i kall på println()
```

Kapittel 10.10

For å illustere poenget med grunn og dyp kopiering endrer vi klassen Person slik at navnet lagres i et StringBuilder-objekt. Adressen lagres fremdeles i et String-objekt. Klassen String er som kjent immutabel, og vi kan derfor aldri forandre på et strengobjekt, vi lager i stedet et nytt objekt. Klassen StringBuilder er derimot mutabel. I tillegg lager vi settmetoder for å endre navn og adresse. Klassen Person ser nå slik ut:

```
class Person implements Cloneable {
  private long fnr;
  private StringBuilder navn;
  private String adresse;

  public Person(long startFnr, String startNavn, String startAdr) {
    fnr = startFnr;
    navn = new StringBuilder(startNavn);
    adresse = startAdr;
  }

  public long finnFnr() {
    return fnr;
  }

  public void settNavn(String nyttNavn) {
    navn.replace(0, navn.length(), nyttNavn); // skifter ut innholdet i navneobjektet
  }

  public void settAdresse(String nyAdresse) {
    adresse = nyAdresse; // referansen adresse vil nå peke til et nytt objekt
  }

  public String finnNavn() {
    return navn.toString();
  }

  public String finnAdresse() {
    return adresse;
  }

  public String toString() {
    return "f.nr.: " + fnr + ", navn: " + navn + ", adresse: " + adresse;
  }
```

```
}
```

Oppgave 1

Grunn kopiering kontra dyp kopiering får betydning dersom et objekt inneholder referanser (og ikke bare variabler av primitive datatyper). Grunn kopiering betyr at referansene kopieres, det vil si at kopi og original refererer til de samme objektene. Dyp kopiering betyr at også det som referansene peker til blir kopiert. I siste tilfelle vil originalen og kopien peke til forskjellige (men like) objekter.

Klassen Person har tre objektvariabler, hvorav to referanser. Ved grunn kopiering vil ikke objektene bli kopiert.

Hva skjer dersom vi forandrer på datainnholdet i objektene? Anta at vi har personobjektet p1. Vi lager et objekt p2 som er en *grunn kopi* av p1. Vi skal så forandre på p2 og se hvilke konsekvenser det får for p1. Vi skiller mellom StringBuilder-objektet og String-objektet ((Eksempel på kjøring av program, se oppgave 4):

- StringBuilder-objektet: Metoden settNavn() forandrer på datainnholdet i objektet. Dette får også konsekvenser for p1, på grunn av at navne-referansene i de to objektene peker til det samme StringBuilder-objektet.
- String-objektet: Metoden settAdresse() tar en referanse som argument. adresse settes lik denne referansen, det vil si at adresse i p2 peker til et annet strengobjekt enn den tilsvarende referansen i p1.

Dersom vi foretar en *dyp kopiering*, vil p2 ha sine egne navn- og adresse-objekter. Vi kan dermed endre både navn og adresse i p2 uten at p1 blir berørt, og omvendt.

Oppgave 2

Standardutgaven clone() foretar en grunn kopiering. Metoden er protected i klassen Object.

Oppgave 3

Vi lager følgende lille program:

```
public static void main(String[] args) {
  Person p1 = new Person(123456789456L, "Ingrid Lund", "Storgt 56, 0213 Oslo");
  Person p2 = (Person) p1.clone();
  System.out.println(p2.toString());
}
```

Dersom vi legger main() i en egen klasse, vil vi ikke på vegne av et Person-objekt ha tilgang til den arvede standardutgaven clone(). Årsaken til dette er at denne metoden er arvet fra en klasse i en annen pakke. (Se kapittel 10.6.) I dette tilfellet får vi kompileringsfeilen "Can't access protected method clone in class java.lang.Object. Person is not a subclass of the current class."

Dersom vi legger main() inne i klassen Person gjelder ikke begrensningen foran. Men da får vi kompileringsfeilen (CloneNotSupportedException) på grunn av at klassen Person ikke implementerer interfacet Cloneable.

Oppgave 4

En klasse som vil tilby metoden clone(), må implementere interfacet Cloneable, og også lage sin egen utgave av clone(). Dersom den inne i denne metoden vil kalle den arvede utgaven av clone(), må unntaket CloneNotSupportedException fanges:

```
class Person implements Cloneable {
.....
public Object clone(){  // grunn kopiering
   try {
     return super.clone();
   } catch (CloneNotSupportedException e) {
     return null;  // dette blir resultatet dersom vi ikke har implements Cloneable
   }
 }
```

Vi prøver denne clone()-metoden:

```
class Oppg10_10_4_grunn {
  public static void main(String[] args) {
    Person p1 = new Person(123456789456L, "Ingrid Lund", "Storgt 56, 0213 Oslo");
    Person p2 = (Person) p1.clone();
    System.out.println(p2.toString());  // kontrollutskrift p2

    /* Vi forandrer dataene i p1 */
    p1.settNavn("Eva Jensen");
    p1.settAdresse("Heia");

    /* Og vi skriver ut p2 */
    System.out.println(p2.toString());
  }
}

/* Utskrift:
f.nr.: 123456789456, navn: Ingrid Lund, adresse: Storgt 56, 0213 Oslo
f.nr.: 123456789456, navn: Eva Jensen, adresse: Storgt 56, 0213 Oslo
*/
```

Vi ser at endringen av navnet ble overført til p2-objektet.

Vi lager en clone()-metode med dyp kopiering:

```
public Object clone(){  // dyp kopiering
  try {
    Person kopi = (Person) super.clone();
    kopi.navn = new StringBuilder(navn.toString());
    return kopi;
  } catch (CloneNotSupportedException e) {
    return null;
  }
}
```

Nå vil ikke endringer i p1 påvirke p2. Mer om clone(), se [Bloch 2001].