Løsning på småoppgaver etter hvert underkapittel, kapittel 13–18

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

Tilpasset 4.utgave av boka.

Kapittel 13.1

```
import static javax.swing.JOptionPane.*;
class Oppg13_1 {
  public static void main(String[] args) {

    /* 1 */
    java.util.ArrayList<Vare> varer = new java.util.ArrayList<Vare>();
    int varenr = 100;
    String varenavn = showInputDialog(null, "Oppgi varenavn (avslutt med Esc): ");
    while (varenavn != null) {
      varenavn = varenavn.trim();
      String prisLest = showInputDialog(null, "Oppgi pris uten moms: ");
      double pris = Double.parseDouble(prisLest);
      Vare enVare = new Vare(varenavn, varenr, pris);
      varer.add(enVare);
      varenr++;
      varenavn = showInputDialog(null, "Oppgi varenavn (avslutt med Esc): ");
    }

    /* 2 */
    String resultat = "";
    for (int i = 0; i < varer.size(); i++) {
      Vare enVare = varer.get(i);
      resultat += (enVare.toString() + "\n");
    }
    showMessageDialog(null, resultat);

    /* 3 */
    String søkenavn = showInputDialog(null,
                          "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.getVarenavn();
      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, "Alle opplysninger om denne varen: "
                            + varen.toString());
        } 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.toString() + "\n");
        }
        showMessageDialog(null, resultat);
      }
    }
```

Kapittel 13.2

Oppgave 1

```
class Oppg13_2_1 {
  public static void main(String[] args) {
    java.util.ArrayList<Flate> flater = new java.util.ArrayList<Flate>();
    Flate f1 = new Flate("F1", 3, 4);
    Flate f2 = new Flate("F2", 2, 4);
    Flate f3 = new Flate("F3", 2, 2);
    flater.add(f1);
    flater.add(f2);
    flater.add(f3);
    double sum = 0.0;
    for (Flate enFlate : flater) {
      sum += enFlate.beregnAreal();
    }
    System.out.println("Sum areal er lik " + sum);
  }
}
```

Oppgave 2

```
/**
 * Metoden finner maksimalnedbøren i måneden.
 */
public int finnMaksimum() {
  int maks = 0;
```

```
    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;
  }

  /**
   * Metoden finner gjennomsnittlig nedbør i måneden rundet av til nærmeste heltall.
   * Returnerer -1 hvis antall dager er 0.
   */
  public int finnGjSnitt() {
    int sum = 0;
    for (int nedbørDag : nedbør) {
      sum += nedbørDag;
    }
    if (nedbør.length > 0) {
      double snitt = (double) sum / (double) nedbør.length;
      return (int) (snitt + 0.5);
    } else {
      return -1;
    }
  }

  /**
   * Metoden finner antall dager uten nedbør.
   */
  public int finnAntTørreDager() {
    int antall = 0;
    for (int nedbørDag : nedbør) {
      if (nedbørDag == 0) {
        antall++;
      }
    }
    return antall;
  }
```

Kapittel 13.3

Oppgave 1

```
class Oppg13_3_1 {
  public static void main(String[] args) {
    java.util.ArrayList<Character> liste = new java.util.ArrayList<Character>();
    String tekst = javax.swing.JOptionPane.showInputDialog(null, "Skriv en tekst: ");
    for (int i = 0; i < tekst.length(); i++) {
```

```
      liste.add(tekst.charAt(i));
    }

    for (char tegn : liste) {
      System.out.print(tegn);
    }
    System.out.println();
  }
}
```

Kapittel 13.5

Oppgave 1

Nye metoder:

```
/**
 * Finner totalt antall studenter på alle fagene.
 */
public int finnTotAntStud() {
  int sum = 0;
  for (Fag f : fagene) {
    sum += f.getAntStud();
  }
  return sum;
}

/**
 * Finner gjennomsnittlig antall studenter pr. fag.
 * Hvis antall fag er 0, returneres 0.
 */
public double finnGjsnittAntStud() {
  int sum = finnTotAntStud();
  if (sum == 0) {  // dekker også antall fag lik 0
    return 0;
  } else {
    return (double) sum / (double) fagene.size();
  }
}

/**
 * Finner hvilke fag som har mer enn et bestemt antall studenter.
 */
public ArrayList<Fag> finnFag(int antStud) {
  ArrayList<Fag> fag = new ArrayList<Fag>();
  for (Fag f : fagene) {
    if (f.getAntStud() > antStud) {
      fag.add(f);
    }
  }
  return fag;
}
```

Testklient:

```
public static void main(String[] args) {
  System.out.println("Totalt antall tester: 4");
  Fagkatalog2 kat = new Fagkatalog2();
  if (kat.finnTotAntStud() == 0 && kat.finnGjsnittAntStud() == 0.0
   && kat.finnFag(0).size() == 0 && kat.finnFag(10).size() == 0) {
     System.out.println("Oppgave 13_5_1: Test1 vellykket");
  }

  /*
  * Bruker ferdig utprøvd metoder til å registrere fag, ingen studenter.
  */
  kat.registrerNyttFag("LC191D", "Videregående programmering");
  kat.registrerNyttFag("LV172D", "Programmering i Java");
  kat.registrerNyttFag("LO347D", "Web-applikasjoner");
  kat.registrerNyttFag("LV172D", "Programmering i Java");
  kat.registrerNyttFag("LC331D", "IT, miljø og samfunn");
  if (kat.finnTotAntStud() == 0 && kat.finnGjsnittAntStud() == 0.0
   && kat.finnFag(0).size() == 0 && kat.finnFag(10).size() == 0) {
     System.out.println("Oppgave 13_5_1: Test2 vellykket");
  }

  /*
  * Bruker ferdig utprøvd metode til å registrere studenter.
  */
  final double TOLERANSE = 0.0001;
  kat.oppdaterAntStud("LC191D", 30);
  kat.oppdaterAntStud("LC331D", 30);
  if (kat.finnTotAntStud() == 60
    && Math.abs(kat.finnGjsnittAntStud() - 15.0) < TOLERANSE
    && kat.finnFag(30).size() == 0) {
      System.out.println("Oppgave 13_5_1: Test3 vellykket");
  }


  ArrayList<Fag> liste = kat.finnFag(25);
  if (liste.size() == 2 && liste.get(0).getFagkode().equals("LC191D")
                        && liste.get(1).getFagkode().equals("LC331D")) {
    System.out.println("Oppgave 13_5_1: Test4 vellykket");
  }
}
```

Kapittel 13.6

Oppgave 1

```
public boolean fjernFlate(String flatenavn) {
  for (int i = 0; i < flatene.size(); i++) {
    Flate denne = flatene.get(i);
    if ((denne.getNavn()).equals(flatenavn)) {
      flatene.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 < malingene.size();
        malingsIndeks++) {
     Maling denneMaling = malingene.get(malingsIndeks);
     if ((denneMaling.getNavn()).equals(malingnavn)) {

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

Klientprogram:

```
  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.setMalingstype(m1);
    f2.setMalingstype(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". */
```

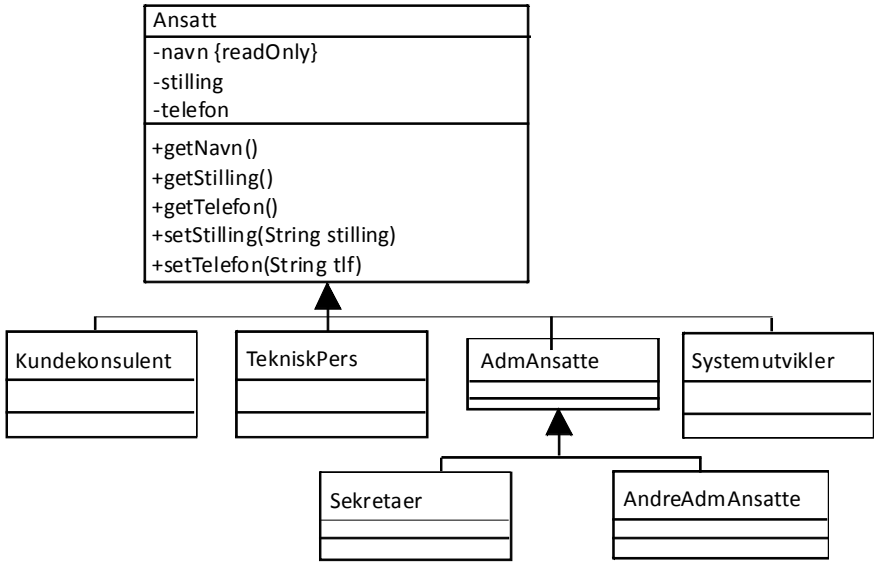
```
    f2.setMalingstype(m1);

    /* 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.getNavn());
    }

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

Utskriften fra dette programmet 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 14.1

Oppgave a)



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 b)

Klassene Kundekonsulent, TekniskPers, AdmAnsatte, Sekretaer, AndreAdmAnsatte og Systemutvikler arver følgende metoder fra klassen Ansatt: getNavn(), getStilling(), getTelefon(), setStilling(), setTelefon().

Kapittel 14.2

Oppgave 1

```
class Ekstern extends Person {
  private final String funksjon;
  private double lønnHittil = 0.0;

  public Ekstern(long fnr, String navn, String adresse, String funksjon) {
    super(fnr, navn, adresse);
    this.funksjon = funksjon;
  }

  public String getFunksjon() {
    return funksjon;
  }
```

```
  public double getLønnHittil() {
    return lønnHittil;
  }

  public void endreLønnHittil(double endring) {
    lønnHittil += endring;
  }

  public String toString() {
    java.util.Formatter f = new java.util.Formatter();
    f.format("%.2f", lønnHittil);
    return "Eksternt tilknyttet person med funksjon: " + funksjon +
          ", lønn hittil kr. " + f.toString() + " " + super.toString();
  }
}

class Oppg14_2_1 {
  public static void main(String[] args) {
    Ekstern eksternPerson = new Ekstern(12106078756L, "Ole Pettersen",
      "Storgt 3, 7001 Trondheim", "timelærer");
    System.out.println(
        "Prøver get-metoder: " +  eksternPerson.getFnr() + " " +
        eksternPerson.getNavn() + " " + eksternPerson.getAdresse() + " " +
        eksternPerson.getFunksjon() + " " + eksternPerson.getLønnHittil());
    eksternPerson.endreLønnHittil(2000);
    eksternPerson.endreLønnHittil(4000);
    System.out.println("Prøver toString(): " + eksternPerson);
  }
}
```

Oppgave 2

Klassen Ansatt er mutabel, stilling og telefon kan endres.

Vi koder subklassene Kundekonsulent og AdmAnsatte med sine subklasser.

Vi koder ikke klassene Systemutvikler og TekniskPers, de blir helt tilsvarende klassen Kundekonsulent.

```
abstract class Ansatt {
  private final String navn;
  private String stilling;
  private String telefon;
  public Ansatt(String navn, String stilling, String telefon) {
    this.navn = navn;
    this.stilling = stilling;
    this.telefon = telefon;
  }

  public String getNavn() {
    return navn;
  }
```

```
  public String getStilling() {
    return stilling;
  }

  public String getTelefon() {
    return telefon;
  }

  public void setStilling(String stilling) {
    this.stilling = stilling;
  }

  public void setTelefon(String telefon) {
    this.telefon = telefon;
  }

  public double beregnLønn() {
    System.out.println("Lønn regnes ut i følge tabell");
    return 0.0;
  }

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

class Kundekonsulent extends Ansatt {
  public Kundekonsulent(String navn, String stilling, String telefon) {
    super(navn, stilling, telefon);
  }

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

class AdmAnsatte extends Ansatt {
  public AdmAnsatte(String navn, String stilling, String telefon) {
    super(navn, stilling, telefon);
  }
  /* Lager ikke toString()-metode her, utgaven arvet fra Ansatt gjelder derfor. */
}

class Sekretaer extends AdmAnsatte {
  public Sekretaer(String navn, String stilling, String telefon) {
    super(navn, stilling, telefon);
  }

  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 navn, String stilling, String telefon) {
     super(navn, stilling, telefon);
   }

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

Oppgave 3

```
class Oppg14_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.setStilling("Oslokontoret");
    konsulent.setTelefon("45454");
    System.out.println("Kundekonsulenten: " + konsulent.getNavn() + ", "
      + konsulent.getStilling() + ", " + konsulent.getTelefon()
      + ", " + konsulent.beregnLønn());

    sekr.setStilling("Stilling 999");
    sekr.setTelefon("67676");
    System.out.println("Sekretæren: " + sekr.getNavn()
      + sekr.getStilling() + ", " + sekr.getTelefon() + ", " + sekr.beregnLønn());

    ikkeSekr.setStilling("Stilling 777");
    ikkeSekr.setTelefon("888888");
    System.out.println("Ikke-sekretæren: " + ikkeSekr.getNavn() + ", "
      + ikkeSekr.getStilling() + ", " + ikkeSekr.getTelefon()
      + ", " + ikkeSekr.beregnLønn());

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

```
    System.out.println(ikkeSekr);
  }
}

/* Utskrift - merk hvor utskriften fra beregnLønn() kommer
Lønn regnes ut i følge tabell
Kundekonsulenten: Ole Ås, Oslokontoret, 45454, 0.0
Lønn regnes ut i følge tabell
Sekretµren: Anne NilsenStilling 999, 67676, 0.0
Lønn regnes ut i følge tabell
Ikke-sekretµren: Anne Nilsen, Stilling 777, 888888, 0.0
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 14.3

Oppgave 1

I klassen Kundekonsulent:

```
public double beregnLønn() {
  System.out.println("Lønn for kundekonsulent er ennå ikke implementert");
  return 0.0;
}
```

I klassen AdmAnsatte:

```
public double beregnLønn() {
  System.out.println("Lønn for administrativt ansatte er ennå ikke implementert");
  return 0.0;
}
```

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 Oppg14_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 (int i  = 0; i < ansatte.length; i++) {
      System.out.println(ansatte[i].getNavn() + " lønn: " +  ansatte[i].beregnLønn());
    }
  }
```

```
}
```

Oppgave 3

```
class Oppg14_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 a : ansatte) {
      System.out.println(a.getNavn() + " lønn: " +  a.beregnLønn());
    }
  }
}
```

Oppgave 4

for-løkken fra oppgave 3 ser nå slik ut:

```
for (Ansatt a : ansatte) {
  String kategori;
  if (a instanceof Kundekonsulent) {
    kategori = "Kundekonsulent";
  } else if (a instanceof Sekretaer) {
    kategori = "Sekretær";
  } else {
    kategori = "Adm.ansatt, men ikke sekretær";
  }
  System.out.println("Kategori: " + kategori + ", " + a.getNavn()
                     + " lønn: " +  a.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 a : ansatte) {
  String kategori  = a.finnKlassenavn();
  System.out.println("Kategori: " + kategori + ", " + a.getNavn()
                     + " lønn: " +  a.beregnLønn());
}
```

Kapittel 14.4

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 beregnOmkrets();
```

Implementasjon i klassen Kvadrat:

```
public double beregnOmkrets() {
  return 4.0 * side;
}
```

Implementasjon i klassen Trekant:

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

Implementasjon i klassen Sirkel:

```
public double beregnOmkrets() {
  return 2.0 * Math.PI * radius;
}
```

Utvider kroppen i for-løkken i klientprogrammet:

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

Oppgave 3

Denne oppgaven er dessverre helt feilplassert i 1.opptrykk av boka (2009). Den hører hjemme etter kapittel 14.7. Det vil som oppgave 2 side 474.

Kapittel 14.5

Oppgave 1

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

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

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

class Oppg14_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");
  }
}
```

Kapittel 14.6

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 14.7

Oppgave 1

Vi bruker i første omgang klientprogrammet:

```
class Oppg14_7_1 {
  public static void main(String[] args) {
    Tapet t = new Tapet("Soloppgang", 300, 15, 0.6);
    Belegg b = new Belegg("Soloppgang", 570, 3);
    if (t.equals(b)) {
      System.out.println("Objektene er like.");
    } else {
      System.out.println("Objektene er ikke like.");
    }
  }
}
```

Å kjøre det uten å endre noe på koden fra programliste 14.5 gir denne utskriften:

```
Objektene er like.
```

Vi deklarerer equals() i klassen Tapet:

```
public boolean equals(Object detAndre) {
  if (!(detAndre instanceof Tapet)) {
    return false;
  }
  return super.equals(detAndre);
}
```

og i klassen Belegg:

```
public boolean equals(Object detAndre) {
  if (!(detAndre instanceof Belegg)) {
    return false;
  }
  return super.equals(detAndre);
}
```

Å prøve ut dette krever noe mer enn den lille testen foran. Har derfor laget et ordinært testprogram med fem tester:

```
public static void main(String[] args) {
  System.out.println("I alt 5 tester:");
  Tapet t = new Tapet("Soloppgang", 300, 15, 0.6);
  Tapet t2 = new Tapet("Soloppgang", 300, 15, 0.6);
  Tapet t3 = new Tapet("Solnedgang", 300, 15, 0.6);

  Belegg b = new Belegg("Soloppgang", 570, 3);
  Belegg b2 = new Belegg("Soloppgang", 570, 3);
  Belegg b3 = new Belegg("Solnedgang", 570, 3);
```

```
    if (!t.equals(b)) {  // Merk at NOT brukes
      System.out.println("Test 1 vellykket.");
    }
    if (t2.equals(t)) {
      System.out.println("Test 2 vellykket.");
    }
    if (!t3.equals(t)) { // Merk at NOT brukes
      System.out.println("Test 3 vellykket.");
    }
    if (b2.equals(b)) {
      System.out.println("Test 4 vellykket.");
    }
    if (!b3.equals(b)) { // Merk at NOT brukes
      System.out.println("Test 5 vellykket.");
    }
  }
```

Oppgave 2

Dette er oppgaven som i 2009-utgaven står som oppgave 3 side 461. Oppgave a) og b) bør også bytte plass, slik at man bruker kopikonstruktøren til å lage metoden lagKopi() – og ikke omvendt. Hensikten med lagKopi() ser man først når fabrikk-begrepet introduseres på side 485.

Oppgave 2a - Kopikonstruktører

Klassen Materiale:

```
public Materiale(Materiale original) {
  this.navn = original.navn;
  this.pris = original.pris;
}
```

Klassen Maling:

```
public Maling(Maling original) {
  super(original);
  this.antStrøk = original.antStrøk;
  this.antKvmPrLiter = original.antKvmPrLiter;
}
```

Klassen Belegg:

```
public Belegg(Belegg original) {
  super(original);
  this.bredde = original.bredde;
}
```

Klassen Tapet:

```
public Tapet(Tapet original) {
  super(original);
  this.lengdePrRull = original.lengdePrRull;
  this.breddePrRull = original.breddePrRull;
}
```

Klient for å prøve dette:

```
class Oppg14_7_2a {
```

```
  public static void main(String[] args) {
    Maling m = new Maling("DekkerGodt", 150, 1, 12);
    Tapet t = new Tapet("Soloppgang", 300, 15, 0.6);
    Belegg b = new Belegg("HimmelOgHav", 570, 3);

    Maling nyM = new Maling(m);
    Tapet nyT = new Tapet(t);
    Belegg nyB = new Belegg(b);

    System.out.println(nyM);
    System.out.println(nyT);
    System.out.println(nyB);
  }
}
```

Oppgave 2b

Klassen Materiale:

```
  public abstract Materiale lagKopi();
```

Klassen Maling:

```
public Materiale lagKopi() {
  return new Maling(this);
}
```

Klassen Belegg:

```
public Materiale lagKopi() {
  return new Belegg(this);
}
```

Klassen Tapet:

```
public Materiale lagKopi() {
  return new Tapet(getNavn(), getPris(), lengdePrRull, breddePrRull);
}
```

Klient for å prøve dette:

```
class Oppg14_7_2b  {
  public static void main(String[] args) {
    Maling m = new Maling("DekkerGodt", 150, 1, 12);
    Tapet t = new Tapet("Soloppgang", 300, 15, 0.6);
    Belegg b = new Belegg("HimmelOgHav", 570, 3);

    Maling kopiM = (Maling) m.lagKopi();
    Tapet kopiT = (Tapet) t.lagKopi();
    Belegg kopiB = (Belegg) b.lagKopi();

    System.out.println(kopiM);
    System.out.println(kopiT);
    System.out.println(kopiB);
  }
}
```

Kapittel 14.8

Oppgave 1

```
class Oppg14_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, 3);
    materialer[0] = belegg1;
    materialer[1] = belegg2A;
    materialer[2] = belegg2B;
    for (Materiale etMateriale : materialer) {
      System.out.print("Belegg: " + etMateriale.getNavn());
      System.out.println(", materialbehov: "
              + etMateriale.beregnMaterialbehov(golv) + " m, pris kr.: " +
                  etMateriale.beregnTotalpris(golv));
    }
  }
}

/* Utskrift:
Belegg: PrimaVare, materialbehov: 4.0 m, pris kr.: 800.0
Belegg: SekundaVare1, materialbehov: 4.8 m, pris kr.: 480.0
Belegg: SekundaVare2, materialbehov: 9.6 m, pris kr.: 576.0000000000001
*/
```

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()
```

Kapirttel 14.10

Oppgave 1

Begrepene grunn og dyp kopiering er kjent fra tidligere. For å repetere og 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 streng-

objekt, vi lager i stedet et nytt objekt. Klassen StringBuilder er derimot mutabel. I tillegg lager vi set-metoder for å endre navn og adresse (egenskapen final gjelder derme ikke lenger). Klassen Person ser nå slik ut:

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

  public Person(long fnr, String navn, String adresse) {
    this.fnr = fnr;
    this.navn = new StringBuilder(navn);
    this.adresse = adresse;
  }

  public long getFnr() {
    return fnr;
  }

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

  public String getAdresse() {
    return adresse;
  }

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

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

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

Grunn kopiering kontra dyp kopiering kan altså få 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 som referansene peker til, 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 setNavn() 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 setAdresse() 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 endrer hodet i klassen Person:

```
class Person implements Cloneable {
```

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 14.6.) 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 Oppg14_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.

Kapittel 15.1

Oppgave 1

```
/**
 * Metoden leser inn et positivt heltall fra brukeren.
 */
public static int lesPosHeltall(String ledetekst) {
  int tall = 0;
  boolean ok = false;
  String melding = ledetekst;
```

```
  do {
    String tallSomTekst = lesTekst(melding);
    try {
      tall = Integer.parseInt(tallSomTekst);
      if (tall > 0) {
        ok = true;
      }
    } catch (NumberFormatException e) {
      melding = "Du skrev: " + tallSomTekst
        + ".\nDu må skrive et positivt heltall. Prøv på nytt.\n" + ledetekst;
    }
  } while (!ok);
  return tall;
}

/**
 * Metoden leser inn et heltall som må ligge i et bestemt intervall avgrenset av
 * grense1 og grense2. Grensene er inkludert i intervallet.
 * Hvis grense2 > grense1, så byttes disse om.
 */
public static int lesTallIIntervall(String ledetekst, int grense1, int grense2) {
  if (grense1 > grense2) {  // bytter om grensene
    int hjelp = grense1;
    grense1 = grense2;
    grense2 = hjelp;
  }
  int tall = 0;
  boolean ok = false;
  String melding = ledetekst;
  do {
    String tallSomTekst = lesTekst(melding);
    try {
      tall = Integer.parseInt(tallSomTekst);
      if (tall >= grense1 && tall <= grense2) {
        ok = true;
      }
    } catch (NumberFormatException e) {
      melding = "Du skrev: " + tallSomTekst
        + ".\nDu må skrive et tall i intervallet [" + grense1 + ", "
        + grense2 + "]. Prøv på nytt.\n" + ledetekst;
    }
  } while (!ok);
  return tall;
}
```

Kapittel 15.3

Oppgave 1

Utskriften blir som følger:

```
Nå er i = 0
Divisjon med null!
```

```
Nå begynner vi på for-løkke nr. 2:
Nå er i = 0
Divisjon med null!
Nå er i = 1
1
Nå er i = 2
0
Nå er i = 3
0
Nå er i = 4
0
```

Første løkke har try-catch-blokken utenfor løkken. Dette fører til at programkontrollen hopper ut av løkken når ArithmeticException kastes.

Andre løkke har try-catch-blokken rundt løkkekroppen. Dersom divisjon med null inntreffer, skrives meldingen ut, og programkontrollen fortsetter med første setning etter try-catch-blokken, det vil si neste gjennomløp av for-løkken.

Årsaken til at vi får ut 0 når i = 2, 3 og 4, skyldes at resultatet fra en heltallsdivisjon er heltallet uten rest.

Oppgave 2

while-løkken ser slik ut:

```
while(scan.hasNext()) {
  try {
    double tall = scan.nextDouble(); // kan kaste InputMismatchException
    sum += tall;
  } catch (InputMismatchException e) {
    System.out.println("Feil ved omforming til tall.");
    scan.next(); // hopper over neste ord
  }
System.out.println("Summen av tallene er " + sum + ".");
}
```

Kapittel 15.4

Oppgave 1

```
class AntSifferException extends Exception {
  public AntSifferException(int antSifferInn, int antSifferKrav) {
    super("Tallet må være positivt og bestå av " + antSifferKrav
          + ". Tallet var negativt, eller det bestod av " + antSifferInn + " siffer.");
  }
}

public Ansatt(int ansattnr, String navn) throws AntSifferException {
  if (ansattnr < 1000 || ansattnr > 9999) {
    String ansTekst = "" + ansattnr;
    throw new AntSifferException(ansTekst.length(), 4);
  }
  this.ansattnr = ansattnr;
  this.navn = navn;
  this.timelønn = 0.0;
```

```
}
```

Kapittel 15.5

Oppgave 1

Kompileringsfeilen er som følger:

```
non-static method getAnsNr() cannot be referenced from
a static context
```

Vi prøver å utføre en objektmetode uten at vi sier hvilket objekt den skal utføres på. Ja, kanskje det ikke eksisterer objekter av klassen Ansatt i det hele tatt. Ansatt er en "static context". Lovlige operasjoner i denne "static context" er Ansatt.FIRMA og Ansatt.finnSistBrukteAnsNr(). (I tillegg kan vi inne i klassen Ansatt skrive Ansatt.sistBrukteAnsNr eller bare sistBrukteAnsNr. Dette kan vi gjøre både inne i objektmetoder og inne i klassemetoder.)

Kapittel 15.6

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 16.3

Oppgave 1

```
import java.io.*;
class Oppg16_3_1 {
  public static void main(String[] args) throws IOException {
    //final String navnPåInnfil = "mineData.txt";
    //final String navnPåInnfil = "MineData" + File.separator + "mineData.txt";
    final String navnPåInnfil = "MineData/mineData.txt";
    System.out.println("Filnavn: " + navnPåInnfil);

    FileReader leseforbTilFil = new FileReader(navnPåInnfil);
    BufferedReader leseren = new BufferedReader(leseforbTilFil);
    int antall = 0;
    String enLinje = leseren.readLine();
    while(enLinje != null) {
      antall++;
      enLinje = leseren.readLine();
    }
    leseren.close();
    System.out.println("Antall linjer lest: " + antall);
  }
}
```

a) Dersom filen ikke eksisterer kastes FileNotFoundException

d) Filnavnet må endres til: "MineData" + File.separator + "mineData.txt" eller "MineData/mineData.txt".

Kapittel 16.4

Oppgave 1

```
import java.io.*;
class Oppg16_4_1 {
  public static void main(String[] args) throws IOException {
    final String navnPåInnfil = "mineData.txt";
    final String navnPåUtfil = "dineData.txt";
    FileReader leseforbTilFil = new FileReader(navnPåInnfil);
    BufferedReader leseren = new BufferedReader(leseforbTilFil);
```

```
    FileWriter skriveforbTilFil = new FileWriter(navnPåUtfil, false);
    PrintWriter skriveren = new PrintWriter(new BufferedWriter(skriveforbTilFil));

    int antall = 0;
    String enLinje = leseren.readLine();
    while(enLinje != null) {
      antall++;
      enLinje = enLinje.toUpperCase();
      skriveren.println(enLinje);
      enLinje = leseren.readLine();
    }
    leseren.close();
    skriveren.close();
    System.out.println("Antall linjer lest: " + antall);
  }
}
```

Kapittel 16.6

Oppgave 1

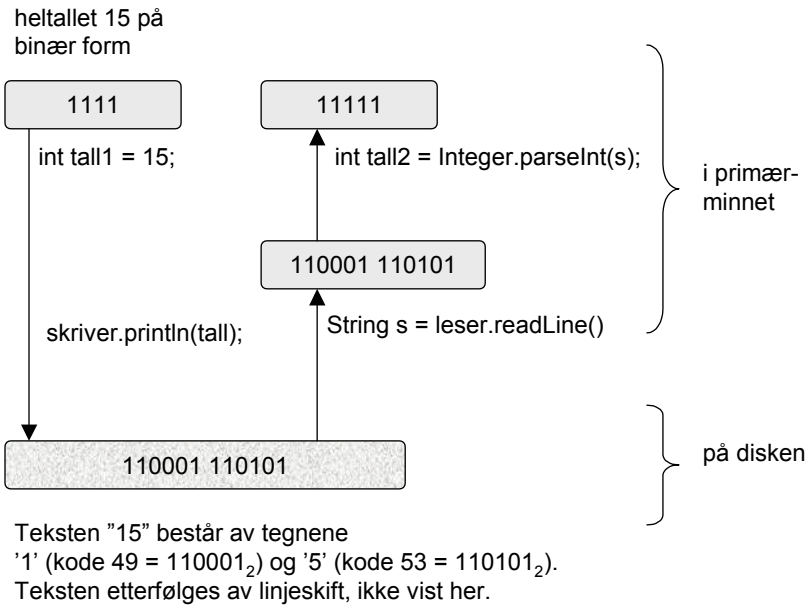
```
import java.util.*;
import java.io.*;
class TallLeser {
  private BufferedReader leseren;
  public TallLeser(BufferedReader leseren) {
    this.leseren = leseren;
  }

  /**
   * Metoden returnerer null dersom linjen inneholder minst et ugyldig heltall.
   * Kaster EOFException dersom ingen data å lese.
   */
  public int[] lesLinjeMedHeltall() throws IOException {
    try {
      String linje = leseren.readLine();
      if (linje != null) {
        StringTokenizer tekst = new StringTokenizer(linje);
        int[] tabell = new int[tekst.countTokens()];
        int indeks = 0;
        while (tekst.hasMoreTokens()) {
          String s = tekst.nextToken();
          tabell[indeks] = Integer.parseInt(s);
          indeks++;
        }
        return tabell;
      } else {
        throw new EOFException("Forsøker å lese en linje, ingen data funnet");
      }
    } catch (NumberFormatException e) {
      return null;
    }
```

```
  }

  /**
   * Metoden returnerer null dersom linjen inneholder minst et ugyldig desimaltall,
   * Kaster EOFException dersom ingen data å lese.
   */
  public double[] lesLinjeMedDesimaltall() throws IOException {
    try {
      String linje = leseren.readLine();
      if (linje != null) {
        StringTokenizer tekst = new StringTokenizer(linje);
        double[] tabell = new double[tekst.countTokens()];
        int indeks = 0;
        while (tekst.hasMoreTokens()) {
          String s = tekst.nextToken();
          tabell[indeks] = Double.parseDouble(s);
          indeks++;
        }
        return tabell;
      } else {
        throw new EOFException("Forsøker å lese en linje, ingen data funnet");
      }
    } catch (NumberFormatException e) {
      return null;
    }
  }
}

class Oppg16_6_1 {
  public static void main(String[] args) throws IOException {
    String filnavn = "tallfil.txt";
    FileReader forbindelseTilFil = new FileReader(filnavn);
    BufferedReader leser = new BufferedReader(forbindelseTilFil);
    TallLeser leseren = new TallLeser(leser);

    /* Bruker en datafil med tre linjer heltall etterfulgt av tre linjer desimaltall. */
    for (int i = 0; i < 3; i++) {
      int[] tall1 = leseren.lesLinjeMedHeltall();
      if (tall1 == null) {
        System.out.println("Linjen inneholder ugyldige heltall");
      } else {
        for (int j = 0; j < tall1.length; j++) {
          System.out.print(tall1[j] + " ");
        }
        System.out.println();
      }
    }
    for (int i = 0; i < 3; i++) {
      double[] tall2 = leseren.lesLinjeMedDesimaltall();
      if (tall2 == null) {
        System.out.println("Linjen inneholder ugyldige desimaltall");
```

```
      } else {
        for (int j = 0; j < tall2.length; j++) {
          System.out.print(tall2[j] + " ");
        }
        System.out.println();
      }
    }
  }
}

/*
Datafil:
1 2 3 4 5.0
67 189 78
5
1.2 3.5 6 78
34.56 56 5.7
4e

Utskrift:
Linjen inneholder ugyldige heltall
67 189 78
5
1.2 3.5 6.0 78.0
34.56 56.0 5.7
Linjen inneholder ugyldige desimaltall
*/
```

Kapittel 16.7

Oppgave 1



Teksten ”15” består av tegnene
’1’ (kode 49 = 110001_2) og ’5’ (kode 53 = 110101_2).
Teksten etterfølges av linjeskift, ikke vist her.

Kapittel 16.8

Oppgave 1

Utskrift vil føre til en blank pr. tegn, det vil si at “ T e k s t” blir til “ T e k s t”. Programmet som finner ut dette ser slik ut:

```
import java.io.*;
class Oppg16_8_1 {
  public static void main(String[] args) {
    String filnavn = "DirektefilTekst.dat";
    try {

      /* Sletter filen dersom den eksisterer */
      File fil = new File(filnavn);  // se API-dokumentasjonen
      fil.delete();

      RandomAccessFile filen = new RandomAccessFile(filnavn, "rw");

      filen.writeChars("Tekst\n");
      System.out.println("Lengde: " + filen.length());
      filen.seek(0);
      String s = filen.readLine();
      System.out.println(s);
      filen.seek(0);
      filen.writeChars(s);
      filen.seek(0);
      s = filen.readLine();
      System.out.println(s);
    } catch (Exception e) {
      System.out.println("Feil oppstått: " + e);
    }
  }
}
```

Dersom vi kjører programmet flere ganger mot den samme filen, vil resultatene bli forskjellige hver gang. Programmet begynner derfor med å slette filen dersom den eksisterer.

Oppgave 2

Foran hver gruppe må vi lagre et tall som forteller antall tall i gruppen.

Kapittel 16.9

Oppgave 1a)

Klassen Vare må implementere interfacet java.io.Sreializable. Det vil si at klassehodet må se slik ut:

```
class Vare implements java.io.Serializable {
```

Oppgave 1b)

Følgende program lager tabellen, fyller den med data, og skriver tabellen som ett objekt til en fil. Deretter leses innholdet i denne filen inn og skrives ut på skjermen:

```
public static void main(String[] args) throws Exception {
  Vare[] tab1 = new Vare[5];
  tab1[0] = new Vare("ost", 100, 70);
  tab1[1] = new Vare("pølse", 101, 60);
  tab1[2] = new Vare("banan", 102, 9.50);
```

```
    tab1[3] = new Vare("agurk", 103, 9.50);
    tab1[4] = new Vare("tomat", 104, 18.50);

    FileOutputStream utstrøm = new FileOutputStream("varer.ser");
    ObjectOutputStream objektskriveren = new ObjectOutputStream(utstrøm);
    objektskriveren.writeObject(tab1);
    objektskriveren.close();

    FileInputStream innstrøm = new FileInputStream("varer.ser");
    ObjectInputStream objektleseren = new ObjectInputStream(innstrøm);
    Vare[] tab2 = (Vare[]) objektleseren.readObject();
    objektleseren.close();

    for (int i = 0; i < tab2.length; i++) {
      System.out.println(tab2[i]);
    }
  }
```

Utskriften blir:

```
100: ost, pris pr. kg kr 70,00 u.moms.
101: pølse, pris pr. kg kr 60,00 u.moms.
102: banan, pris pr. kg kr 9,50 u.moms.
103: agurk, pris pr. kg kr 9,50 u.moms.
104: tomat, pris pr. kg kr 18,50 u.moms.
```

Av dette kan vi slutte at tabellobjekter kan serialiseres.

Oppgave 2

Dette programmet lager et tabellobjekt og serialiserer det. Deretter leses det inn fra filen og skrives ut.

```
  public static void main(String[] args) throws Exception {
    int[] tab1 = {1, 12, 45, 34, 56};
    FileOutputStream utstrøm = new FileOutputStream("tall.ser");
    ObjectOutputStream objektskriveren = new ObjectOutputStream(utstrøm);
    objektskriveren.writeObject(tab1);
    objektskriveren.close();

    FileInputStream innstrøm = new FileInputStream("tall.ser");
    ObjectInputStream objektleseren = new ObjectInputStream(innstrøm);
    int[] tab2 = (int[]) objektleseren.readObject();
    objektleseren.close();

    for (int i = 0; i < tab2.length; i++) {
      System.out.println(tab2[i]);
    }
  }
```

Kapittel 17.2

Oppgave 1-4

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

class TrykknappVindu extends JFrame {
  public TrykknappVindu(String tittel) {
    setTitle(tittel);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    LayoutManager layout = new FlowLayout();
    setLayout(layout);

    JButton knapp = new JButton("Trykk her!!");  // lager knappen
    add(knapp);  // legger den i beholderen
    Knappelytter knappelytteren = new Knappelytter(); // lager en lytter
    knapp.addActionListener(knappelytteren); // knytter lytteren til knappen

     /* Oppgave 1 */
    JButton knapp2 = new JButton("Eller trykk her!!");
    add(knapp2);
    knapp2.addActionListener(knappelytteren);

    /* Oppgave 2
     * For at Alt+T, ev. Alt+E, skal virke må knappen ha fokus,
     * tastaturalternativet for å få fokus er tab-tasten.
     */
    knapp.setMnemonic('T');
    knapp2.setMnemonic('E');

    /* Oppgave 3 */
    knapp2.setEnabled(false); // nå kan ikke brukeren trykke på knapp 2

    pack();  // tilpasser vindusstørrelsen
  }
}

class Knappelytter implements ActionListener {
  public void actionPerformed(ActionEvent hendelse) {
    /* Oppgave 4, endrer utskriftsetningen */
    String tekst = hendelse.getActionCommand();
    System.out.println("Du trykket på knapp med tekst: " + tekst);
  }
}

class Oppg17_2_1til4 {
  public static void main(String[] args) {
    TrykknappVindu etVindu = new TrykknappVindu("Et vindu med en knapp");
    etVindu.setVisible(true);
  }
}
```

Kapittel 17.3

Oppgave 1

Følgende setning legges til slutt i metoden actionPerformed():

```
navnefelt.setEditable(false);
```

Oppgave 2

Vi må nå telle antall trykk pr. knapp. Både knapper og tellere må være objektvariabler:

```
private JButton knappRød = new JButton("Rød");
private JButton knappBlå = new JButton("Blå");
private JButton knappTurkis = new JButton("Turkis");
private int antTrykkRød = 0;
private int antTrykkBlå = 0;
private int antTrykkTurkis = 0;
private static final int MAKSTRYKK = 3;
```

Husk å fjerne deklarasjonene av knappene fra konstruktøren!

Metoden actionPerformed() ser nå slik ut:

```
public void actionPerformed(ActionEvent hendelse) {
  JButton valgtKnapp = (JButton) hendelse.getSource();
  String fargenavn = valgtKnapp.getText();
  Color farge;
  if (fargenavn.equals("Rød")) {
    farge = Color.RED;
    antTrykkRød++;
    if (antTrykkRød > MAKSTRYKK) {
      knappRød.setEnabled(false);
    }
  } else if (fargenavn.equals("Blå")) {
    farge = Color.BLUE;
    antTrykkBlå++;
    if (antTrykkBlå > MAKSTRYKK) {
      knappBlå.setEnabled(false);
    }
  } else { // turkis
    farge = Color.cyan;
    antTrykkTurkis++;
    if (antTrykkTurkis > MAKSTRYKK) {
      knappTurkis.setEnabled(false);
    }
  }
  hilsen.setForeground(farge);
  String navn = navnefelt.getText();
  hilsen.setText("Hei på deg, " + navn + "!");
}
```

Oppgave 3

```
import java.awt.*;
```

```
import java.awt.event.*;
import javax.swing.*;

class NavneVindu3Knapp extends JFrame {
  private JTextField navnefelt = new JTextField(20);
  private JLabel hilsen = new JLabel("Her kommer en hilsen til deg");

  private JLabel ledetekst = new JLabel("Skriv navnet ditt:");
  private Font sansSerif;
  private Font monoSpaced;
  private Font dialog;
  private JButton knappSansSerif = new JButton("SansSerif");
  private JButton knappMonoSpaced = new JButton("MonoSpaced");
  private JButton knappDialog = new JButton("Dialog");

  public NavneVindu3Knapp(String tittel) {
    setTitle(tittel);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    setLayout(new FlowLayout());

    ledetekst = new JLabel("Skriv navnet ditt:");
    add(ledetekst);
    add(navnefelt);

    /* Oppretter tre forskjellige skrifttyper */
    Font standardskrift = navnefelt.getFont();
    int stil = standardskrift.getStyle();
    int str = standardskrift.getSize();
    sansSerif = new Font("SansSerif", stil, str);
    monoSpaced = new Font("Serif", stil, str);
    dialog = new Font("Dialog", stil, str);

    /* En skrifttype pr knapp */
    knappSansSerif.setFont(sansSerif);
    add(knappSansSerif);

    knappMonoSpaced.setFont(monoSpaced);
    add(knappMonoSpaced);

    knappDialog.setFont(dialog);
    add(knappDialog);

    Knappelytter knappelytteren = new Knappelytter();
    knappSansSerif.addActionListener(knappelytteren);
    knappMonoSpaced.addActionListener(knappelytteren);
    knappDialog.addActionListener(knappelytteren);

    add(hilsen);
    pack();
  }
```

```
    private class Knappelytter implements ActionListener {
      public void actionPerformed(ActionEvent hendelse) {
        /* Endrer skrifttyper i stedet for bakgrunnsfarge */
        JButton valgtKnapp = (JButton) hendelse.getSource();
        String skrifttype = valgtKnapp.getText();
        Font nyFont;
        if (skrifttype.equals("SansSerif")) {
          nyFont = sansSerif;
        } else if (skrifttype.equals("MonoSpaced")) {
          nyFont = monoSpaced;
        } else {
          nyFont = dialog;
        }
        hilsen.setFont(nyFont);
        navnefelt.setFont(nyFont);
        ledetekst.setFont(nyFont);
        /* (endrer ikke skrifttypene på knappene)*/

        String navn = navnefelt.getText();
        hilsen.setText("Hei på deg, " + navn + "!");
      }
    }
  }

  class Oppg17_3_3 {
    public static void main(String[] args) {
      NavneVindu3Knapp etVindu =
                    new NavneVindu3Knapp("Tekster og tre knapper");
      etVindu.setVisible(true);
    }
  }
```

Oppgave 4

Den hendelsen at brukeren trykker på Enter-tasten er en ActionEvent. Vi lager følgende indre lytterklasse:

```
    private class NavnefeltLytter implements ActionListener {
      public void actionPerformed(ActionEvent hendelse) {
        String navn = navnefelt.getText();
        hilsen.setText("Hei på deg, " + navn + "!");
        knappRød.requestFocus();  // knappRød må være objektvariabel
      }
    }
```

Vi registrerer en slik lytter:

```
  navnefelt.addActionListener(new NavnefeltLytter());
```

Kapittel 17.4

Oppgave 1

Parallelle tabeller med farger og fargenavn deklareres som konstanter i klassen VinduMedTegning:

```
private static final String[] ALLE_FARGENAVN =
 {"Svart", "Blå", "Turkis", "Mørkgrå", Grå", "Grønn", "Lysgrå", "Magenta", "Oransje",
  "Rosa", "Rød", "Hvit", "Gul"};
private static final Color[] ALLE_FARGER =
 {Color.BLACK, Color.BLUE, Color.CYAN, Color.DARK_GRAY, Color.GRAY,
  Color.GREEN, Color.LIGHT_GRAY, Color.MAGENTA, Color.ORANGE,
  Color.PINK, Color.RED, Color.WHITE, Color.YELLOW};
```

Avsnittet under kommentaren "Finner fargen til knappen" ser nå slik ut:

```
JButton valgtKnapp = (JButton) hendelse.getSource();
String fargenavn = valgtKnapp.getText();
Color farge = Color.BLACK;
int fargeIndeks = 0;
while (fargeIndeks < ALLE_FARGENAVN.length &&
        !ALLE_FARGENAVN[fargeIndeks].equals(fargenavn)) {
  fargeIndeks++;
}
if (fargeIndeks < ALLE_FARGENAVN.length) {
  farge = ALLE_FARGER[fargeIndeks];
} else {
  System.out.println("Feil oppstått i Knappelytter.");
}
```

Vi henter ut fargenavnet, og deretter går vi gjennom tabellen ALLE_FARGENAVN for å finne indeksen til denne fargen. Vi slår så opp i tabellen ALLE_FARGER på samme indeks. Der finner vi riktig fargeobjekt.

Søndre panel ser nå slik ut:

```
private class Knappepanel extends JPanel {
  public Knappepanel() {
    Knappelytter knappelytteren = new Knappelytter();
   setLayout(new GridLayout(3, 5, 5, 5)); // Tre rader a fem knapper
    /* Vi bruker en løkke til å lage knappene. */
    for (int knappNr = 0; knappNr < ALLE_FARGENAVN.length; knappNr++) {
      JButton knapp = new JButton(ALLE_FARGENAVN[knappNr]);
      knapp.setBackground(ALLE_FARGER[knappNr]);
      knapp.addActionListener(knappelytteren);
      add(knapp);
    }
  }
}
```

Kapittel 18.2

Oppgave 1

Vi må gjøre tillegg flere steder i programmet:

- Ny objektvariabel i klassen AvkrysningsruteVindu:

  ```
  private JCheckBox frokost = new JCheckBox("frokost");
  ```

- Objektvariabelen må legges inn i valgpanelet:

```
add(frokost); // i konstruktøren ValgPanel()
```

- Vi må knytte lytterobjektet til denne avkrysningsruten:

```
frokost.addActionListener(lytter); // i konstruktøren ValgPanel()
```

- I actionPerformed() må vi ta med utskrift til kommandovinduet:

```
if (frokost.isSelected()) {
  System.out.println("Skal ha frokost.");
}
```

Oppgave 2

Vi skal legge inn et tekstområde under gruppeboksen.

- Dette området må metoden actionPerformed() i klassen CheckBoxLytter ha adgang til. Det må derfor være en objektvariabel:

```
private JTextArea utskrift = new JTextArea(3, 30);
```

- Vi legger inn dette tekstområdet sør i vinduet:

```
add(utskrift, BorderLayout.SOUTH);  // i konstruktøren
```

- Metoden actionPerformed() blir helt ny:

```
public void actionPerformed(ActionEvent hendelse) {
  String tekst = "";
  if (middag.isSelected()) {
    tekst += "Skal ha middag.\n";
  }
  if (lunsj.isSelected()) {
    tekst += "Skal ha lunsj.\n";
  }
  utskrift.setText(tekst);
}
```

Kapittel 18.3

Oppgave 1

Vi skal ha to gruppebokser inne i valgpanelet. Det får vi til ved å dele dette panelet i to mindre paneler, et for valg av kjønn og et for valg av bolig. Vi har laget to lytterklasser, vi kunne også greid oss med én. Programmet i sin helhet ser slik ut:

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.border.*;

class RadioknappVindu extends JFrame {
  /* Knappene merket "kvinne" og “hybel” er trykket inn ved programstart. */
  private JRadioButton kvinne = new JRadioButton("kvinne", true);
  private JRadioButton mann = new JRadioButton("mann", false);
  private JRadioButton hybel = new JRadioButton("Hybel", true);
  private JRadioButton leilighet = new JRadioButton("Leilighet", false);
  private JRadioButton rekkehus = new JRadioButton("Rekkehus", false);
  private JRadioButton enebolig = new JRadioButton("Enebolig", false);
```

```
public RadioknappVindu(String tittel) {
  setTitle(tittel);
  setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

  /* Beholderen har samme oppbygning som i TestAvkrysningsruter.java */
  add(new JPanel(), BorderLayout.NORTH); // litt luft
  add(new JPanel(), BorderLayout.SOUTH); // litt luft
  ValgPanel midten = new ValgPanel();
  add(midten, BorderLayout.CENTER);
  pack();
}

private class ValgPanel extends JPanel {
  public ValgPanel() {
    add(new KjønnPanel());
    add(new BoligPanel());
  }
}

private class KjønnPanel extends JPanel {
  public KjønnPanel() {
    ButtonGroup gruppe = new ButtonGroup();
    gruppe.add(kvinne);
    gruppe.add(mann);
    add(kvinne);
    add(mann);

    RadioknappLytter lytter = new RadioknappLytter();
    kvinne.addActionListener(lytter);
    mann.addActionListener(lytter);

    SoftBevelBorder ramme = new SoftBevelBorder(BevelBorder.RAISED);
    Border gruppeboks = BorderFactory.createTitledBorder(ramme, "Kjønn");
    setBorder(gruppeboks);
  }
}

private class BoligPanel extends JPanel {
  public BoligPanel() {
    ButtonGroup gruppeBolig = new ButtonGroup();
    gruppeBolig.add(hybel);
    gruppeBolig.add(leilighet);
    gruppeBolig.add(rekkehus);
    gruppeBolig.add(enebolig);
    add(hybel);
    add(leilighet);
    add(rekkehus);
    add(enebolig);
    RadioknappLytterBolig lytterBolig = new RadioknappLytterBolig();
    hybel.addActionListener(lytterBolig);
    leilighet.addActionListener(lytterBolig);
```

```
      rekkehus.addActionListener(lytterBolig);
      enebolig.addActionListener(lytterBolig);
      SoftBevelBorder ramme = new SoftBevelBorder(BevelBorder.RAISED);
      Border gruppeboks = BorderFactory.createTitledBorder(ramme, "Bolig");
      setBorder(gruppeboks);
    }
  }

  /* Lytterobjekter som lytter til alle endringer i radioknappene */
  private class RadioknappLytter implements ActionListener {
    public void actionPerformed(ActionEvent hendelse) {
      String kjønn = hendelse.getActionCommand();
      if (kjønn.equals("kvinne")) {
        System.out.println("Kvinne er valgt");
      } else {
        System.out.println("Mann er valgt");
      }
    }
  }

  private class RadioknappLytterBolig implements ActionListener {
    public void actionPerformed(ActionEvent hendelse) {
      String bolig = hendelse.getActionCommand();
      System.out.println(bolig + " er valgt.");
    }
  }
}

class Oppg18_3_1 {
  public static void main(String[] args) {
    RadioknappVindu etVindu = new RadioknappVindu("Kjønn og bolig");
    etVindu.setVisible(true);
  }
}
```

Oppgave 2

Vi legger trykknappen i konstruktøren RadioknappVindu() og registrerer en lytter til knappen:

```
JButton knapp = new JButton("Registrer valg");
add(knapp, BorderLayout.SOUTH);
knapp.addActionListener(new Knappelytter());
```

Husk å fjerne det tomme panelet som ligger i sør.

Lytterne fjernes fra radioknappene (i konstruktøren ValgPanel(), linje 42–44).

Lytterklassen ser nå slik ut:

```
  private class Knappelytter implements ActionListener {
    public void actionPerformed(ActionEvent hendelse) {
      if (kvinne.isSelected()) {
        System.out.println("Kvinne er valgt");
```

```
      } else {
        System.out.println("Mann er valgt");
      }
    }
  }
```

Kapittel 18.4

Oppgave 1

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;

class ListeboksVindu extends JFrame {
  private static final String [] BYER =
    {"Arendal", "Bergen", "Bodø", "Gjøvik", "Hamar", "Hammerfest", "Kristiansund",
     "Moss", "Stavanger", "Tromsø", "Trondheim", "Ålesund"};
  private static final int[] FOLKETALL =
    {39247, 227276, 40767, 26968, 26424, 9151,
    16928, 26242, 108019, 58121, 147187, 38251}; // oppg. 1
  private JTextField tekst = new JTextField("Du har ennå ikke valgt byer.     ");
  private JList byliste = new JList(BYER);  // Nå er listen laget!

  public ListeboksVindu(String tittel) {
    setTitle(tittel);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    JLabel ledetekst = new JLabel("Velg en by");
    add(ledetekst, BorderLayout.NORTH);

    /* Legger på rullefelt */
    JScrollPane rullefeltMedListe = new JScrollPane(byliste);
    add(rullefeltMedListe, BorderLayout.CENTER);

    ListeboksLytter lytter = new ListeboksLytter();
    byliste.addListSelectionListener(lytter);

     byliste.setSelectionMode(ListSelectionModel.SINGLE_SELECTION); // oppg.1

    tekst.setEditable(false); // brukeren skal ikke kunne editere i feltet
    add(tekst, BorderLayout.SOUTH);
    pack();
  }

  /* Lytteren fanger opp alle klikk på linjer i listeboksen */
  class ListeboksLytter implements ListSelectionListener {
     public void valueChanged(ListSelectionEvent hendelse) {  // ny i oppg 1
       int valgtIndeks = byliste.getSelectedIndex();
        if (valgtIndeks >= 0) {
       JOptionPane.showMessageDialog(null, "Det bor " + FOLKETALL[valgtIndeks]
```

```
            + " innbyggere i " + BYER[valgtIndeks] + ".");
          byliste.clearSelection();
        }
      }
    }
  }

  class Oppg18_4_1 {
    public static void main(String[] args) {
      ListeboksVindu etVindu = new ListeboksVindu("Valg av byer");
      etVindu.setVisible(true);
    }
  }
```

Oppgave 2

Valget "Ny by" legges inn som en trykknapp under listen i stedet for det teksfeltet som ligger der nå. Bynavnene lagres i et objekt av klassen DefaultListModel, mens folketallene lagres i en ArrayList. Husk at nye byer skal legges på alfabetisk riktig plass i listen.

```
  import java.awt.*;
  import java.awt.event.*;
  import javax.swing.*;
  import javax.swing.event.*;
  import java.util.ArrayList;
  import java.text.Collator;
  import static javax.swing.JOptionPane.*;

  class ListeboksVindu extends JFrame {
    private static final String [] BYER =
      {"Arendal", "Bergen", "Bodø", "Gjøvik", "Hamar", "Hammerfest", "Kristiansund",
       "Moss", "Stavanger", "Tromsø", "Trondheim", "Ålesund"};
    private static final int[] FOLKETALL = {
       39247, 227276, 40767, 26968, 26424, 9151,
      16928, 26242, 108019, 58121, 147187, 38251}; // oppg. 1
    private JTextField tekst = new JTextField("Du har ennå ikke valgt byer.     ");

    private DefaultListModel listeinnhold = new DefaultListModel();
    private JList byliste = new JList(listeinnhold);  // Obs!
    private ArrayList<Integer> folketallene = new ArrayList<Integer>();
    private Collator kollator = Collator.getInstance(); // til sortering av byene

    public ListeboksVindu(String tittel) {
      setTitle(tittel);
      setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

      /* Diverse initieringer */
      for (int i = 0; i < BYER.length; i++) {
        listeinnhold.addElement(BYER[i]);
        folketallene.add(FOLKETALL[i]); // auto-boxing
      }
```

```
  JLabel ledetekst = new JLabel("Velg en by");
  add(ledetekst, BorderLayout.NORTH);

  /* Legger på rullefelt */
  JScrollPane rullefeltMedListe = new JScrollPane(byliste);
  add(rullefeltMedListe, BorderLayout.CENTER);

  ListeboksLytter lytter = new ListeboksLytter();
  byliste.addListSelectionListener(lytter);

  byliste.setSelectionMode(ListSelectionModel.SINGLE_SELECTION); // oppg.1

 /* Vi har tatt bort tekstfeltet under listen. I stedet har vi lagt inn en trykknapp der. */
  JButton nyByKnapp = new JButton("Ny by");
  add(nyByKnapp, BorderLayout.SOUTH);
  nyByKnapp.addActionListener(new NyByKnappelytter());
  pack();
}

/* Lytteren fanger opp alle klikk på linjer i listeboksen */
class ListeboksLytter implements ListSelectionListener {
   public void valueChanged(ListSelectionEvent hendelse) {  // ny i oppg 1
     int valgtIndeks = byliste.getSelectedIndex();
     if (valgtIndeks >= 0) {
       showMessageDialog(null, "Det bor " + folketallene.get(valgtIndeks)
         + " innbyggere i " + listeinnhold.get(valgtIndeks) + ".");
       byliste.clearSelection();
     }
  }
}

class NyByKnappelytter implements ActionListener {
  public void actionPerformed(ActionEvent hendelse) {
    String nyBy = showInputDialog(null, "Ny by: ");
    if (nyBy != null) {
      int posNyBy = 0;  // Byen skal på sortert plass i listen, finner først plassen
      while (posNyBy < listeinnhold.size()
            && kollator.compare((String) (listeinnhold.get(posNyBy)), nyBy) < 0) {
          posNyBy++;
      }
      listeinnhold.add(posNyBy, nyBy);
      folketallene.add(posNyBy, lesNyttFolketall());
    }
  }

 /* Hjelpemetode: Leser inn tallet. Spør på nytt dersom det ikke er et positivt tall. */
  private Integer lesNyttFolketall() {
    boolean okTall = true;
    Integer nyttFolketall = new Integer(0);
    do {
      String tallSomTekst = showInputDialog(null, "Folketall: ");
```

```
        try {
          nyttFolketall = new Integer(tallSomTekst);
          if (nyttFolketall.intValue() <= 0) {
            throw new NumberFormatException("");
          }
          okTall = true;
        } catch (NumberFormatException e) {
          okTall = false;
          showMessageDialog(null,
            "Det du skrev inn kan ikke tolkes som et positivt tall. Prøv igjen.");
        }
      } while (!okTall);
      return nyttFolketall;
    }
  }
}

class Oppg18_4_2 {
  public static void main(String[] args) {
    ListeboksVindu etVindu = new ListeboksVindu("Valg av byer");
    etVindu.setVisible(true);
  }
}
```

Kapittel 18.5

Oppgave 1

Bytt ut linjen

```
private SpinnerModel spinnerInnhold = new SpinnerListModel(mnd);
```

med

```
private SpinnerNumberModel spinnerInnhold =
          new SpinnerNumberModel(50, 10, 100, 10);
```

Da vil verdien 50 vises i det spinneren kommer opp.

I tillegg bør selvfølgelig variabelnavnene og tekstene i programmet reflektere at vi har tallverdier og ikke måneder.

Kapittel 18.6

Oppgave 1

Vi kaller klassen Tekstvindu i stedet for Tallvindu. Antall ord legges inn som en klassekonstant i denne klassen.

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.text.*;
import javax.swing.event.*;

class Tekstvindu extends JFrame {
  private final int ANT_ORD = 3;
```

```
  private JButton knapp = new JButton("Du har ennå ikke skrevet noe");

  public Tekstvindu(String tittel) {
    setTitle(tittel);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    JLabel ledetekst = new JLabel("Skriv tre ord:");
    add(ledetekst, BorderLayout.NORTH);
    JTextField tekstfelt = new JTextField(20);
    add(tekstfelt, BorderLayout.CENTER);
    tekstfelt.setInputVerifier(new TekstKontroll());
    add(knapp, BorderLayout.SOUTH);
    pack();
  }

  private class TekstKontroll extends InputVerifier {
    private int antOrdFunnet;
    private int[] ordlengder = new int[ANT_ORD];

    public boolean verify(JComponent inndata) {
      JTextField data = (JTextField) inndata;
      String tekst = data.getText();
      java.util.StringTokenizer ordene = new java.util.StringTokenizer(tekst, " ");
      boolean ok = false;
      antOrdFunnet = ordene.countTokens();
      if (antOrdFunnet == ANT_ORD) {
        ok = true;
        for (int i = 0; i < ANT_ORD; i++) {
          String etOrd = ordene.nextToken();
          ordlengder[i] = etOrd.length();
        }
      }
      return ok;
    }

    public boolean shouldYieldFocus(JComponent inndata) {
      boolean ok =
            super.shouldYieldFocus(inndata);  // må kalle superklassens metode
      String tekst;
      if (ok) {
        tekst = "Ordlengder: ";
        for (int i = 0; i < ANT_ORD - 1; i++) tekst += ordlengder[i] + ", ";
        tekst += ordlengder[ANT_ORD - 1];
      } else {
        tekst = "Du har skrevet " + antOrdFunnet + " ord.";
      }
      knapp.setText(tekst);
      return ok;
    }
  }
}
```

```
class Oppg18_6_1 {
  public static void main(String[] args) {
    Tekstvindu etVindu = new Tekstvindu("Firesifret tall");
    etVindu.setVisible(true);
  }
}
```

Kapittel 18.7

Oppgave 1

Legg inn setningen tekstfelt.selectAll(); rett etter setningen tekstfelt.setEditable(true); i metoden focusLost() i klassen Fokuslytter.