Løsning på småoppgaver etter hvert underkapittel. kap. 11-18

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 11.2

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.");
  }
System.out.println("Summen av tallene er " + sum + ".");
}
```

Kapittel 11.3

Oppgave 1

Konstruktøren ser nå slik ut:

```
public Ansatt(int startNr, String startNavn, double startLønn)
                                throws IllegalArgumentException {
  NumberFormat pengeformat = NumberFormat.getCurrencyInstance();
  if ((startNr < nedreGrenseNr || startNr > øvreGrenseNr) && startLønn < minLønn) {
          throw new IllegalArgumentException(
                  "Både ansattnummer og lønn er ugyldig." +
          "\nLønn: " + pengeformat.format(startLønn) +
           ", den skal være minst " + pengeformat.format(minLønn) +
           "\nAnsattnummer: " + startNr + ", det må ligge i intervallet [" +
                   nedreGrenseNr + ", " + øvreGrenseNr + "].");
  } else if (startLønn < minLønn) {
   throw new IllegalArgumentException("Lønn: " + pengeformat.format(startLønn) +
          "\nLønnen skal være minst " + pengeformat.format(minLønn));
  } else if (startNr < nedreGrenseNr || startNr > øvreGrenseNr) {
    throw new IllegalArgumentException("Nummer: " + startNr +
                "\nAnsattnummeret må ligge i intervallet [" +
                  nedreGrenseNr + ", " + øvreGrenseNr + "].");
  } else {
    ansNr = startNr;
    navn = startNavn;
    lønn = startLønn;
  }
}
```

Kapittel 12.3

Oppgave 1

```
import java.io.*;
class Oppg12_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 12.4

Oppgave 1

```
import java.io.*;
class Oppg12_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 12.6

Oppgave 1

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

  /*
   * 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 Oppg12_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
*/
```

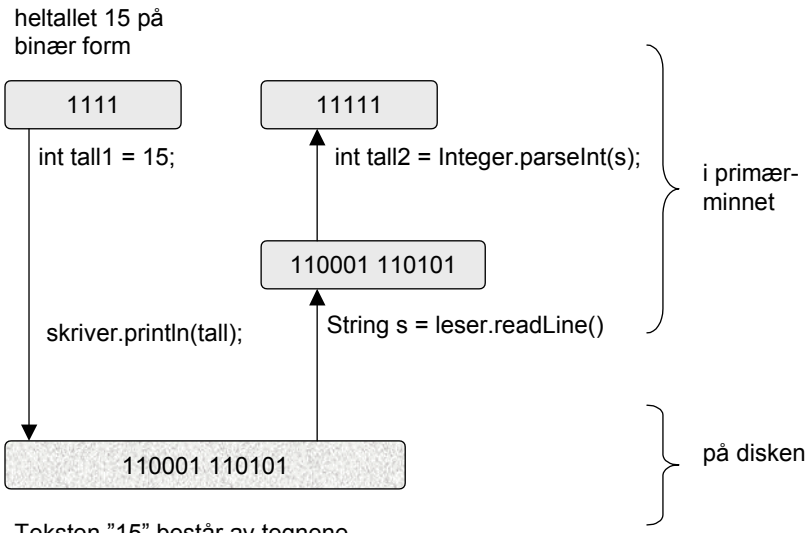
Kapittel 12.7

Oppgave 1

```
import java.util.*;
import java.io.*;
class Oppg12_7_1 {
  public static void main(String[] args) throws Exception {
    InputStreamReader leseforbTilKonsoll = new InputStreamReader(System.in);
    BufferedReader leseren = new BufferedReader(leseforbTilKonsoll);
    System.out.print("Skriv noen heltall, avslutt med CTRL+Z på egen linje: ");
    System.out.flush();   // tømmer bufferet etter at print() er brukt
    int sum = 0;
    Scanner scan = new Scanner(leseren);
    while (scan.hasNext()) {
      sum += scan.nextInt();
    }
    System.out.println("Summen av tallene er " + sum);
  }
}
```

Kapittel 12.8

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 12.9

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 Oppg12_9_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 12.10

Oppgave 1

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

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

Oppgave 2

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 (Vare v : tab2) {
      System.out.println(v.finnNavn() + ", " + v.finnVarenr() + ", " + v.finnPris());
    }
  }
```

Utskriften blir:

```
ost, 100, 70.0
pølse, 101, 60.0
banan, 102, 9.5
agurk, 103, 9.5
tomat, 104, 18.5
```

Av dette kan vi slutte at tabellobjekter kan serialiseres.

Oppgave 3

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 tall : tab2) {
    System.out.println(tall);
  }
```

Kapittel 13.2

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 Oppg13_2_1til4 {
  public static void main(String[] args) {
    TrykknappVindu etVindu = new TrykknappVindu("Et vindu med en knapp");
    etVindu.setVisible(true);
  }
}
```

Kapittel 13.3

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;
```

Metoden actionPerformed() ser nå slik ut:

```
public void actionPerformed(ActionEvent hendelse) {
  String fargenavn = hendelse.getActionCommand();
  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 */
    String skrifttype = hendelse.getActionCommand();
    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 Oppg13_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 13.4

Oppgave 1

De viktigste endringene skjer i metoden actionPerformed() i klassen Knappelytter. Avsnittet under kommentaren "Finner fargen til knappen" ser nå slik ut:

```
JButton knappTrykketPå = (JButton) hendelse.getSource();
Color farge;
if (knappTrykketPå == knappRød) farge = Color.red;
else if (knappTrykketPå == knappBlå) farge = Color.blue;
else farge = Color.cyan;
```

Vi ser at vi refererer til knappe-objektene. Disse kan derfor ikke lenger være lokale variabler i konstruktøren Knappepanel(), men må være objektvariabler i den ytre klassen VinduMedTegning. På denne måten nåes de fra alle metoder og klasser deklarert inne i klassen VinduMedTegning. Setningene

```
JButton knappRød = new JButton("Rød");
```

```
JButton knappBlå = new JButton("Blå");
JButton knappTurkis = new JButton("Turkis");
```

flyttes derfor fra konstruktøren Knappepanel() til *foran* konstruktøren i den ytre klassen. Dessuten får de det reserverte ordet private foran seg:

```
private JButton knappRød = new JButton("Rød");
private JButton knappBlå = new JButton("Blå");
private JButton knappTurkis = new JButton("Turkis")
```

Oppgave 2

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

```
private static final String[] alleFargenavn = {"Svart", "Blå", "Turkis", "Mørkgrå",
  "Grå", "Grønn", "Lysgrå", "Magenta", "Oransje", "Rosa", "Rød", "Hvit", "Gul"};

private static final Color[] alleFarger =
  {Color.black, Color.blue, Color.cyan, Color.darkGray, Color.gray, Color.green,
    Color.lightGray, Color.magenta, Color.orange, Color.pink, Color.red,
     Color.white, Color.yellow};
```

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

```
String fargenavn = hendelse.getActionCommand();
Color farge = Color.black;
int fargeIndeks = 0;
while (fargeIndeks < alleFargenavn.length &&
  !alleFargenavn[fargeIndeks].equals(fargenavn)) fargeIndeks++;
if (fargeIndeks < alleFargenavn.length) farge = alleFarger[fargeIndeks];
else System.out.println("Feil oppstått i Knappelytter.");
```

Vi henter ut fargenavnet, og deretter går vi gjennom tabellen alleFargenavn for å finne indeksen til denne fargen. Vi slår så opp i tabellen alleFarger 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 < alleFargenavn.length; knappNr++) {
      JButton knapp = new JButton(alleFargenavn[knappNr]);
      knapp.setBackground(alleFarger[knappNr]);
      knapp.addActionListener(knappelytteren);
      add(knapp);
    }
  }
}
```

Kapittel 14.2

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 konsollet:

```
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 14.3

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 en. 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 AvkrysningsruteViser.java */
  add(new JPanel(), BorderLayout.NORTH); // litt luft
  add(new JPanel(), BorderLayout.SOUTH); // litt luft
  ValgPanel midten = new ValgPanel();
  guiBeholder.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();
    if (bolig.equals("Hybel")) System.out.println("Hybel er valgt");
    else if (bolig.equals("Leilighet")) System.out.println("Leilighet er valgt");
    else if (bolig.equals("Rekkehus")) System.out.println("Rekkehus er valgt");
    else System.out.println("Enebolig er valgt");
  }
 }
}

class Oppg14_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(), de tre linjene etter add(mann);).

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 14.4

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();
       /*
        * Vi bruker getValueIsAdjusting() slik at vi behandler kun en hendelse
        * pr valg. Hvis vi ikke gjør det vil meldingsboksen
        * vises flere ganger. Dette ser vi ikke fordi boksene kommer oppe på
        * hverandre. Men vi må trykke på OK-knappen flere ganger.
        * Vi må også kontrollere at valgtIndeks > 0
        * pga at clearSelection() (nederst) sender meldingen valueChanged()
        */
       if (!byliste.getValueIsAdjusting() && valgtIndeks >= 0) {
```

```
            JOptionPane.showMessageDialog(null, "Det bor " + folketall[valgtIndeks]
              + " innbyggere i " + byer[valgtIndeks] + ".");
            byliste.clearSelection();
          }
        }
      }
    }

    class Oppg14_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 tabell-liste. Husk at nye byer må 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, se kap.
    9.7

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

        /* Diverse initieringer */
        for (String enBy : byer) listeinnhold.addElement(enBy);
        for (int antPers : folketall) folketallene.add(antPers); // auto-boxing, se s. 331

        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();
       /*
        * Vi bruker getValueIsAdjusting() slik at vi behandler kun en melding
        * pr valg. Hvis vi ikke gjør det vil meldingsboksen
        * vises flere ganger. Dette ser vi ikke fordi boksene kommer oppe på
        * hverandre. Men vi må trykke på OK-knappen flere ganger.
        * Vi må også kontrollere at valgtIndeks > 0
        * pga at clearSeleection() (nederst) sender meldingen valueChanged()
        */
       if (!byliste.getValueIsAdjusting() && 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());  // folketallet på samme plass
som byen
      }
    }
```

```
    /* 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 Oppg14_4_2 {
  public static void main(String[] args) {
    ListeboksVindu etVindu = new ListeboksVindu("Valg av byer");
    etVindu.setVisible(true);
  }
}
```

Kapittel 14.5

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 14.6

Oppgave 1

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

```
private final int antOrd = 3;
```

Klassen for kontroll av data ser nå slik ut:

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

  public boolean verify(JComponent inndata) {
    JTextField data = (JTextField) inndata;
    String tekst = data.getText();
    StringTokenizer ordene = new StringTokenizer(tekst, " ");
    boolean ok = false;
    antOrdFunnet = ordene.countTokens();
    if (antOrdFunnet == antOrd) {
      ok = true;
      for (int i = 0; i < antOrd; 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 < antOrd - 1; i++) tekst += ordlengder[i] + ", ";
      tekst += ordlengder[antOrd - 1];
    } else {
      tekst = "Du har skrevet " + antOrdFunnet + " ord.";
    }
    knapp.setText(tekst);
    return ok;
  }
}
```

Kapittel 14.7

Oppgave 1

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

Kapittel 15.1

Oppgave 1

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

class VinduMedMeny extends JFrame {
  private static final String[] alleFargenavn = {"Svart", "Blå", "Turkis", "Mørkgrå",
```

```
      "Grå", "Grønn", "Lysgrå", "Magenta", "Oransje", "Rosa", "Rød", "Hvit", "Gul"};
  private static final Color[] alleFarger =
    {Color.black, Color.blue, Color.cyan, Color.darkGray,
    Color.gray, Color.green, Color.lightGray, Color.magenta, Color.orange,
    Color.pink, Color.red, Color.white, Color.yellow};
  private Container guiBeholder;

  public VinduMedMeny(){
   setTitle("Menytest");
   setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
   guiBeholder = getContentPane();

   MenyLytter lytteren = new MenyLytter();
   JMenu menyen = new JMenu("Farge");
   JMenuItem[] menyposter = new JMenuItem[alleFargenavn.length];
   for (int i = 0; i < menyposter.length; i++) {
    JMenuItem menypost = new JMenuItem(alleFargenavn[i]);
    menyen.add(menypost);
    menypost.addActionListener(lytteren);
   }
   JMenuBar menylinje = new JMenuBar();
   menylinje.add(menyen);
   setJMenuBar(menylinje);
  }

  private class MenyLytter implements ActionListener {
   public void actionPerformed(ActionEvent hendelse) {
    String kommando = hendelse.getActionCommand();
    int fargeIndeks = 0;
    while (fargeIndeks < alleFargenavn.length
      && !kommando.equals(alleFargenavn[fargeIndeks])) fargeIndeks++;
    if (fargeIndeks < alleFargenavn.length) {
      guiBeholder.setBackground(alleFarger[fargeIndeks]);
     } else System.out.println("Feil oppstått i actionPerformed()"); // skal ikke skje!
   }
  }
 }

 class Oppg15_1_1 {
  public static void main(String[] args) {
   VinduMedMeny vindu = new VinduMedMeny();
   vindu.setSize(300, 200); // ellers vil det ikke vises på skjermen
   vindu.setVisible(true);
  }
 }
```

Oppgave 2

Tillegg i konstruktøren VinduMedMeny():

```
   Hjelpelytter hjelpelytteren = new Hjelpelytter();
   JMenu hjelpemeny = new JMenu("Hjelp");
   menypost = new JMenuItem("Instruksjoner");
```

```
menypost.addActionListener(hjelpelytteren);
hjelpemeny.add(menypost);
menypost = new JMenuItem("Om programmet");
menypost.addActionListener(hjelpelytteren);
hjelpemeny.add(menypost);
menylinje.add(hjelpemeny);
```

En ny lytterklasse (kunne også utvidet den eksisterende):

```
private class Hjelpelytter implements ActionListener {
  public void actionPerformed(ActionEvent hendelse) {
    String kommando = hendelse.getActionCommand();
    if (kommando.equals("Instruksjoner")) {
       showMessageDialog(null,
             "Trekk ned farge-menyen og velg bakgrunnsfarge!");
    } else {
       showMessageDialog(null,
            "Dette programmet er sist revidert 18.november 2004.");
    }
  }
}
```

Kapittel 15.2

Oppgave 1

Legg inn setningen knapperad.addSeparator(); etter for eksempel setningen knapperad.add(gulKnapp); Du vil da se at det blir et bredere mellomrom mellom de to første knappene enn mellom knapp to og tre.

Oppgave 2

Eksempel på hvordan man knytter tooltips til en knapp:

```
knapp.setToolTipText("Nå blir bakgrunnsfargen rød");
```

Dersom brukeren holder musa over knappen en liten stund, kommer en gul etikett med denne teksten fram.

Oppgave 3

For den gule knappen ser kildekoden nå slik ut (inkl. endringene fra oppgave 1 og 2):

```
gulKnapp = new JButton("Gul");
gulKnapp.addActionListener(lytteren);
gulKnapp.setToolTipText("Nå blir bakgrunnsfargen gul");
gulKnapp.setMnemonic('G');
knapperad.add(gulKnapp);
knapperad.addSeparator();
```

Kapittel 15.4

Oppgave 1

Vi har gjennomgått tre lytterinterface:

- Interfacet java.awt.event.ActionListener inneholder bare én metode, og har derfor ingen adapterklasse.

- Interfacet java.awt.event.FocusListener inneholder to metoder. Ettersom interfacet tilhører pakken java.awt har den en adapterklasse, klassen FocusAdapter.
- Interfacet javax.swing.event.ListSelectionListener inneholder bare én metode. Det tilhører dessuten ikke pakken java.awt, og det er dermed to grunner til at dette interfacet ikke har noen adapterklasse.

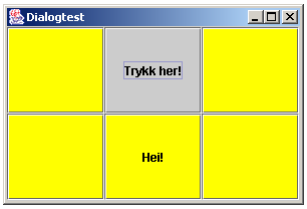
Vi endrer eksemplet i programliste 14.7 slik at vi får prøvd klassen FocusAdapter:

```
private class Fokuslytter extends FocusAdapter {  // eneste linje som endres
  public void focusLost(FocusEvent hendelse) {
    ... som tidligere ...
  }
}
/* Metoden focusGained() skal være tom og kan dermed taes bort */
```

Kapittel 15.5

Oppgave 1

Vi lager et vindu som ser slik ut etter at brukeren har skrevet inn en tekst ("Hei!") og trykket på knappen:



Det består av seks trykknapper, hvorav alle unntatt den det står "Trykk her!" på er gule. De er objekter av klassen GulKnapp. Her er hele programmet:

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

class MiniDialog extends JDialog {
  private boolean ok;
  private JTextField tekstfelt = new JTextField(15); // oppgaven

  public MiniDialog(JFrame forelder) {
    super(forelder, "Minidialog", true);
    add(new JLabel(), BorderLayout.NORTH);
    add(new TekstPanel(), BorderLayout.CENTER);  // oppgaven
    add(new KnappePanel(), BorderLayout.SOUTH);
    pack();
  }

  private class TekstPanel extends JPanel {  // ny klasse i oppgaven
    public TekstPanel() {
      add(new JLabel("Skriv en tekst: "));
```

```
      add(tekstfelt);
    }
  }

  private class KnappePanel extends JPanel {
    ... som før ...
  }

  private class KnappelytterDialog implements ActionListener {
    ... som før ...
  }

  public String visDialog() {  // revidert i oppgaven
    setVisible(true);
    if (ok) return tekstfelt.getText();
    else return null;
  }
}

/*
 * Foreldrevinduet til dialogen. Dette vinduet inneholder kun én knapp.
 */
class ForeldreVindu extends JFrame {
  private MiniDialog dialogboks = new MiniDialog(this);
  private GulKnapp teksten = new GulKnapp();  // endret

  public ForeldreVindu() {  // mye forandret i oppgaven
    setTitle("Dialogtest");
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setLayout(new GridLayout(2, 3));
    add(new GulKnapp());
    JButton trykkKnapp = new JButton("Trykk her!");
    add(trykkKnapp);
    KnappeLytterForeldre knappelytter = new KnappeLytterForeldre();
    trykkKnapp.addActionListener(knappelytter);
    add(new GulKnapp());
    add(new GulKnapp());
    add(teksten);
    add(new GulKnapp());
    pack();
  }

  class GulKnapp extends JButton {  // ny i oppgaven, en gul knapp
    public GulKnapp() {
      setBackground(Color.yellow);
    }
  }

  private class KnappeLytterForeldre implements ActionListener {
    public void actionPerformed(ActionEvent hendelse) {  // endret
      String tekst = dialogboks.visDialog();
```

```
      if (tekst != null) teksten.setText(tekst);
      else teksten.setText("Ikke noe registrert");
    }
  }
}

class Oppg15_5_1 {
  static public void main(String[] args) {
    ForeldreVindu foreldrevindu = new ForeldreVindu();
    foreldrevindu.setSize(300, 200);
    foreldrevindu.setVisible(true);
  }
}
```

Kapittel 15.6

Oppgave 1

I stedet for å ta dataene direkte inn i JTable-objektet

```
private JTable bytabell = new JTable(byer, kolonnenavn);
```

må vi nå legge dem inn i et DefaultTableModel-objekt. I utgangspunktet vil dette objektet ikke inneholde data:

```
private DefaultTableModel datamodell = new DefaultTableModel(kolonnenavn, 0);
private JTable bytabell = new JTable(datamodell);
```

Vi legger inn opplysninger om de byene vi har i konstruktøren:

```
datamodell.addRow(lagRad("Ny by", ""));  // øverst i tabellen
for (int i = 1; i < byer.length; i++) {
  datamodell.addRow(byer[i]);   // rad med indeks i, i den to-dim. tabellen byer
}
```

Metoden addRow() skal som argument ha en tabell av referanser som peker til de objektene som skal vises i raden. I øverste linje viser vi "Ny by" i venstre kolonne. lagRad() er en hjelpemetode som konstruerer en rad ut fra to strenger:

```
private Object[] lagRad(String verdi1, String verdi2) {
  Object[] rad = new Object[2];
  rad[0] = verdi1;
  rad[1] = verdi2;
  return rad;
}
```

Metoden valueChanged() ser nå slik ut:

```
public void valueChanged(ListSelectionEvent hendelse) {
  int linje = bytabell.getSelectedRow();
  if (linje == 0) { // Ny by
    String nyttBynavn = showInputDialog("Skriv bynavnet: ");
    String folketall = showInputDialog("Skriv folketallet: ");
    /* Kontrollerer ikke at et gyldig tall ble skrevet inn */
    datamodell.addRow(lagRad(nyttBynavn, folketall));
  } else if (linje > 0) {
    valget.setText("Du har nå valgt " + bytabell.getValueAt(linje, 0) +
            " med " + bytabell.getValueAt(linje, 1) + " innbyggere.");
```

```
  }
}
```

Kapittel 15.8

Oppgave 1

Vi holder klassen FlateDialog utenfor. Vi lar de tre øvrige dialogklassene få en felles superklasse MaterialDialog:

```
/* Klassen MaterialDialog er felles superklasse for alle materialdialogene */
abstract class MaterialDialog extends MinDialog {
  private NumberFormat innformat = NumberFormat.getNumberInstance();
  private JTextField navneFelt = new JTextField(Konstanter.tekstfeltLengde);
  private JTextField prisFelt = new JTextField(Konstanter.tekstfeltLengde);
  private double pris = 0.0;

  public MaterialDialog(JFrame foreldre, String tittel) {
    super(foreldre, tittel);
    add(new JPanel(), BorderLayout.NORTH);
    add(finnKnappepanel(), BorderLayout.SOUTH);
  }

  protected NumberFormat finnInnformat() {
    return innformat;
  }

  protected double finnPris() {
    return pris;
  }

  protected String finnNavn() {
    return navneFelt.getText();
  }

  /*
   * Hver enkelt dialogklasse vil lage sin egen subklasse til klassen Datapanel.
   * Et eksempel på at en indre klasse med fordel kan være noe annet enn privat...
   */
  protected class Datapanel extends JPanel {
    public Datapanel(String ledetekstNavn, String enhet, int antLinjer) {
      setLayout(new GridLayout(4, antLinjer));
      add(new JLabel(ledetekstNavn));
      add(navneFelt);
      add(new JLabel("Pris pr " + enhet + " :"));
      add(prisFelt);
    }
  }

  /*
   * Metoden okData() deklareres ikke i subklassene. I stedet inneholder de
   * en metode tolkRestenAvTallDataene(). Denne metoden er abstrakt her,
   * se nedenfor.
```

```
     */
    final protected boolean okData() {
      try {
        pris = innformat.parse(prisFelt.getText()).doubleValue();
        tolkRestenAvTalldata();
      } catch (ParseException e) {
        showMessageDialog(null, "Tall-input kan ikke tolkes. Prøv igjen.");
        prisFelt.requestFocus();
        return false;
      }
      return true;
    }

    /*
     * Hver enkelt subklasse har sin egen visDialog(). Den "nullstiller" de
     * feltene som er aktuelle for den klassen, deretter kalles denne
     * utgaven av visDialog().
     */
    public Materiale visDialog() {
      navneFelt.setText("");
      prisFelt.setText("");
      settOk(false);
      setVisible(true);
      navneFelt.requestFocus();

      /* Se hver enkelt subklasses implementasjon av metoden lagNyttMateriale() */
      if (erOk()) return lagNyttMateriale();
      else return null;
    }

    /* To abstrakte metoder i denne klassen. De er forklart i kommentarene over. */
    abstract protected void tolkRestenAvTalldata() throws ParseException;
    abstract protected Materiale lagNyttMateriale();
  }
```

Som eksempel på subklasse tar vi med klassen MalingDialog:

```
  class MalingDialog extends MaterialDialog {
    private JTextField antStrøkFelt = new JTextField(Konstanter.tekstfeltLengde);
    private JTextField antKvmPrLFelt = new JTextField(Konstanter.tekstfeltLengde);
    private int antStrøk;
    private double antLiter;

    public MalingDialog(JFrame foreldre) {
      super(foreldre, "Maling");
      add(new MalingDatapanel(), BorderLayout.CENTER);
      pack();
    }

    private class MalingDatapanel extends Datapanel {
      public MalingDatapanel() {
        super("Malingnavn: ", "liter", 4);
```

```
      add(new JLabel("Antall strøk: "));
      add(antStrøkFelt);
      add(new JLabel("Antall kvm pr liter: "));
      add(antKvmPrLFelt);
    }
  }

  protected void tolkRestenAvTalldata() throws ParseException {
    antStrøk = finnInnformat().parse(antStrøkFelt.getText()).intValue();
    antLiter = finnInnformat().parse(antKvmPrLFelt.getText()).doubleValue();
  }

  protected Materiale lagNyttMateriale() {
    return new Maling(finnNavn(), finnPris(), antStrøk, antLiter);
  }

  public Materiale visDialog() {
    antStrøkFelt.setText("");
    antKvmPrLFelt.setText("");
    return super.visDialog();
  }
}
```

På klientsiden i klassen OppussingKap15GUI må vi gjøre en liten endring i den private klassen Knappelytter. Metoden visDialog() har (og må ha) returtype Materiale. Enten må vi "caste" til riktig klasse, eller vi må la returverdien også tilhøre denne typen. Enten må vi skrive:

```
case 1:  // ny maling
  Maling nyMaling = (Maling) malingDialogen.visDialog();
  registrerMateriale(nyMaling);
  break;
```

eller:

```
case 1:  // ny maling
  Materiale nyMaling = malingDialogen.visDialog();
  registrerMateriale(nyMaling);
  break;
```

Begge variantene vil også fungere i det opprinnelige programmet.

Kapittel 15.9

Oppgave 1

Bruk av egen datamodell tvinger oss til å være mer ryddig i forhold til dataene. Vi lager en klasse ByInfo som inneholder informasjon om en by:

```
class ByInfo {
  private String bynavn;
  private String folketall;

  public ByInfo(String startBynavn, String startFolketall) {
    bynavn = startBynavn;
```

```
    folketall = startFolketall;
  }

  public String finnBynavn() {
    return bynavn;
  }

  public String finnFolketall() {
    return folketall;
  }
}
```

Dataene våre består av informasjon om mange byer. Vi har behov for å hente ut verdiene til en by, og å legge inn nye ByInfo-objekter. Vi velger å bruke klassen ArrayList som tilfredsstiller disse kravene. Vi oppretter et objekt av denne klassen i main(). Vi fyller det med startdata.

Klassen TabellVindu skal nå i så stor grad som mulig konsentrere seg om brukergrensesnittet.

Konstruktøren til TabellVindu tar i mot dataobjektet fra main() og sender det videre til datamodellobjektet som danner forbindelsen mellom brukergrensesnittet og dataene. Klassen med main() ser slik ut:

```
class Oppg15_9_1 {
  static final String [][] byer = {{"Arendal", "39247"}, {"Bergen", "227276"},
  {"Bodø", "40767"}, {"Gjøvik", "26968"}, {"Hamar", "26424"},
  {"Hammerfest", "9151"}, {"Kristiansund", "16928"},
  {"Moss", "26242"}, {"Stavanger", "108019"},   {"Tromsø", "58121"},
  {"Trondheim", "147187"}, {"Ålesund", "38251"}};

  public static void main(String[] args) {
    ArrayList<ByInfo> mangeByer = new ArrayList<ByInfo>();
    mangeByer.add(new ByInfo("Ny by", ""));
    for (int i = 0; i < byer.length; i++) {
      mangeByer.add(new ByInfo(byer[i][0], byer[i][1]));
    }

    TabellVindu etVindu = new TabellVindu(mangeByer, "Valg av by");
    etVindu.setVisible(true);
  }
}
```

Her er klassen TabellVindu:

```
class TabellVindu extends JFrame {
  private JLabel valget = new JLabel("Du har ennå ikke valgt byer.");
  private ArrayList<ByInfo> byene;  // dataene
  private JTable bytabell;  // presentasjonen av dataene
  private ByDataModell datamodell;  // bindeledd mellom presentasjon og data
  public TabellVindu(ArrayList<ByInfo> mangeByer, String tittel) {
    byene = mangeByer;
    datamodell = new ByDataModell(byene);
    bytabell = new JTable(datamodell);
```

```
    setTitle(tittel);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

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

    /* Legger tabellen inn i et rullefelt */
    JScrollPane rullefeltMedTabell = new JScrollPane(bytabell);
    add(rullefeltMedTabell, BorderLayout.CENTER);

    /* Setter størrelsen på vinduet tabellen vises i */
    bytabell.setPreferredScrollableViewportSize(new Dimension(300, 100));

    /* Setter valgmodellen slik at brukeren kun kan velge én linje av gangen */
    bytabell.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);

    /* Lytteren må kobles til den tilhørende valgmodellen. */
    ListSelectionModel linjevalg = bytabell.getSelectionModel();
    LinjeLytter lytter = new LinjeLytter();
    linjevalg.addListSelectionListener(lytter);

    add(valget, BorderLayout.SOUTH);
    pack();
  }

  /* Lytteren fanger opp alle klikk på linjer i tabellen. */
  class LinjeLytter implements ListSelectionListener {
    public void valueChanged(ListSelectionEvent hendelse) {
      int linje = bytabell.getSelectedRow();
      if (linje == 0) { // Ny by
        String nyttBynavn = showInputDialog("Skriv bynavnet: ");
        String folketall = showInputDialog("Skriv folketallet: ");
        /* Kontrollerer ikke at et gyldig tall ble skrevet inn */
        byene.add(new ByInfo(nyttBynavn, folketall));
        datamodell.fireTableDataChanged();
      } else if (linje > 0) {
        valget.setText("Du har nå valgt " + bytabell.getValueAt(linje, 0) +
                  " med " + bytabell.getValueAt(linje, 1) + " innbyggere.");
      }
    }
  }
}
```

Her er vår egen datamodellklasse:

```
class ByDataModell extends AbstractTableModel {
  private ArrayList<ByInfo> bydata;
  public ByDataModell(ArrayList<ByInfo> startBydata) {
    bydata = startBydata;
  }

  public String getColumnName(int indeks) {
    if (indeks == 0) return "By";
```

```
    else return "Folketall";
  }

  public boolean isCellEditable(int rad, int kolonne) {
    return false;
  }

  public int getColumnCount() {
    return 2;
  }

  public int getRowCount() {
    return bydata.size();
  }

  public Object getValueAt(int rad, int kolonne) {
    ByInfo enBy = bydata.get(rad);
    if (kolonne == 0) return enBy.finnBynavn();
    else return enBy.finnFolketall();
  }
}
```

Kapittel 16.3

Oppgave 1

```
class Skriver implements Runnable {
  private int tallSomSkrives;

  public Skriver(int tall) {
    tallSomSkrives = tall;
  }

  public void run() {
    while (true) {
      System.out.print(tallSomSkrives);
      System.out.print(" ");
    }
  }
}

class Tallskur {

  public static void main(String[] args) {
    Thread skriver1 = new Thread(new Skriver(1));
    Thread skriver2 = new Thread(new Skriver(2));
    Thread skriver3 = new Thread(new Skriver(3));
    Thread skriver4 = new Thread(new Skriver(4));
    Thread skriver5 = new Thread(new Skriver(5));
    skriver1.start();
    skriver2.start();
    skriver3.start();
```

```
    skriver4.start();
    skriver5.start();
  }
}
```

Kapittel 16.6

Oppgave 1

```
class SekvensSkriver {

  public void skrivSekvens() {
    synchronized (this) {
      System.out.print("1 ");
      System.out.print("2 ");
      System.out.print("3 ");
      System.out.print("4 ");
      System.out.print("5 ");
    }
  }
}

class Tallskriver extends Thread {
  private SekvensSkriver skriveren;

  public Tallskriver(SekvensSkriver skriver) {
    skriveren = skriver;
  }

  public void run() {
    while (true) {
      skriveren.skrivSekvens();
    }
  }
}

class TallskurSynk {
  public static void main(String[] args) {
    SekvensSkriver enSkriver = new SekvensSkriver();
    Tallskriver skriver1 = new Tallskriver(enSkriver);
    Tallskriver skriver2 = new Tallskriver(enSkriver);
    Tallskriver skriver3 = new Tallskriver(enSkriver);
    Tallskriver skriver4 = new Tallskriver(enSkriver);
    Tallskriver skriver5 = new Tallskriver(enSkriver);
    skriver1.start();
    skriver2.start();
    skriver3.start();
    skriver4.start();
    skriver5.start();
  }
}
```

Kapittel 17.2

Oppgave 3

Element-klassen blir slik:

```
class NavneElement {
  private NavneElement neste, forrige;
  private String navn;

  public NavneElement(String startNavn) {
    navn = startNavn;
    neste = null;
    forrige = null;
  }

  public NavneElement(String startNavn, NavneElement startNeste,
        NavneElement startForrige) {
    navn = startNavn;
    neste = startNeste;
    forrige = startForrige;
  }

  public String finnNavn() {
    return navn;
  }

  public NavneElement finnNeste() {
    return neste;
  }

  public NavneElement finnForrige() {
    return forrige;
  }

  public void settNavn(String nyttNavn) {
    navn = nyttNavn;
  }

  public void settNeste(NavneElement nyNeste) {
    neste = nyNeste;
  }

  public void settForrige(NavneElement nyForrige) {
    neste = nyForrige;
  }
}
```

Og den nye NavneListe blir slik:

```
class NavneListe {
  private NavneElement førsteElement;

  /* Metoden returnerer innholdet i listen i tekstlig form. */
  public String toString() {
```

```
  String listeTekst = "";
  NavneElement hjelpeReferanse = førsteElement;
  while (hjelpeReferanse != null) {
    listeTekst = listeTekst + hjelpeReferanse.finnNavn() + "\n";
    hjelpeReferanse = hjelpeReferanse.finnNeste();
  }
  return listeTekst;
}

public void settInnNavnTilSlutt(String nyttNavn) {
  NavneElement hjelpeReferanse = førsteElement;
  NavneElement hjelpeReferanseRettBak = null;
  /* Vi starter med å bevege oss fram til slutten av listen. */
  while (hjelpeReferanse != null) {
    hjelpeReferanseRettBak = hjelpeReferanse;
    hjelpeReferanse = hjelpeReferanse.finnNeste();
  }
  if (hjelpeReferanseRettBak == null ) {
    /* Dette skjer om listen var tom fra før. */
    førsteElement = new NavneElement(nyttNavn);
  } else {
    /*
     * Dette skjer dersom det var minst ett element i listen fra
     * før. hjelpeReferanseRettBak refererer nå til det siste
     * elementet.
     */
    hjelpeReferanseRettBak.settNeste(new NavneElement(nyttNavn));
    /* Så setter vi det nyinsatte elementets forrige til å peke
     * bakover.
     */
    hjelpeReferanseRettBak.finnNeste().settForrige(
                                    hjelpeReferanseRettBak.finnForrige());
  }
}

public boolean søkEtterNavn(String søkeTekst) {
  NavneElement hjelpeReferanse = førsteElement;
  while (hjelpeReferanse != null) {
    if (hjelpeReferanse.finnNavn().equals(søkeTekst)) return true;
    hjelpeReferanse = hjelpeReferanse.finnNeste();
  }
  return false;
}

  /*
   * Denne metoden sletter alle forekomster av den gitte teksten fra
   * listen.
   */
public void slettNavn(String navnSomSkalSlettes) {
  NavneElement hjelpeReferanse = førsteElement;
  NavneElement hjelpeReferanseRettBak = null;
```

```
      /* Vi starter med å iterere oss framover i listen. */
      while (hjelpeReferanse != null) {
        if (hjelpeReferanse.finnNavn().equals(navnSomSkalSlettes)) {
            /*
             * Hjelpereferanse refererer nå til et element som skal
             * slettes. Vi må ha spesialbehandling dersom dette er det
             * første elementet i listen.
             */
            if (hjelpeReferanseRettBak != null) {
              hjelpeReferanseRettBak.settNeste(hjelpeReferanse.finnNeste());
            } else {
              førsteElement = hjelpeReferanse.finnNeste();
            }
            hjelpeReferanse = hjelpeReferanse.finnNeste();
            /* Dersom det elementet vi nå har slettet ikke var det siste, må
             * det neste få endret sin forrige-referanse. Den peker jo på den som
             * skal slettes. (Vi anbefaler blyantoppgavene!)
             */
            if (hjelpeReferanse != null) {
              hjelpeReferanse.settForrige(hjelpeReferanseRettBak);
            }
        } else {
            hjelpeReferanseRettBak = hjelpeReferanse;
            hjelpeReferanse = hjelpeReferanse.finnNeste();
        }
      }
    }
  }
```

Testklienten forblir uendret. (Innkapsling!)

Kapittel 17.3

Oppgave 1

```
import java.util.*;

class NavneListeTestLinkedList {

  public static void main(String[] args) {
    LinkedList<String> enListe = new LinkedList<String>();
    System.out.println("Utskrift av tom liste: ");
    System.out.println(enListe.toString());
    enListe.add("Arne");
    enListe.add("Krille");
    System.out.println("Utskrift av to gode kompiser: ");
    /* Utskriften blir ikke helt lik (men enda penere), på grunn av at
     * LinkedList sin toString() er litt anderledes enn vår.
     */
    System.out.println(enListe.toString());
    System.out.println("Finnes Mikke i listen: " + enListe.contains("Mikke"));
    System.out.println("Finnes Arne i listen: " + enListe.contains("Arne"));
```

```
    System.out.println("Finnes Krille i listen: " + enListe.contains("Krille"));
    System.out.println("Setter inn en ekstra Arne...");
    enListe.add("Arne");
    System.out.println("Utskrift med to Arne'er: ");
    System.out.println(enListe.toString());
    System.out.println("Sletter Arne og Krille...");
    while (enListe.contains("Arne")) {
      enListe.remove("Arne");
    }
    while (enListe.contains("Krille")) {
      enListe.remove("Krille");
    }
    System.out.println("Utskrift av tom liste: ");
    System.out.println(enListe.toString());
  }
}

/* Utskrift:
Utskrift av tom liste:
[]
Utskrift av to gode kompiser:
[Arne, Krille]
Finnes Mikke i listen: false
Finnes Arne i listen: true
Finnes Krille i listen: true
Setter inn en ekstra Arne...
Utskrift med to Arne'er:
[Arne, Krille, Arne]
Sletter Arne og Krille...
Utskrift av tom liste:
[]

*/
```

Kapittel 17.4

Oppgave 1

Vi tenker oss vi først tar for oss nevneren. Først gjøres multiplikasjonen 5x89 og svaret lagres på stakken. Så gjøres 9-3, og svaret lagres på stakken. Så gjøres addisjon på de to øverste tallene på stakken. Deretter regnes telleren ut på samme måte, før det gjøres divisjon på de to øverste tallene på stakken.

Kapittel 17.5

Oppgave 2

```
class Palindrom {

  public boolean erPalindrom(String tekst) {
    tekst = tekst.toLowerCase();
    tekst.trim();
    if (tekst.equals("")) return true;
```

```
    if (tekst.length() == 1) return true;
    if (tekst.charAt(0) != tekst.charAt(tekst.length()-1)) return false;
    /* Hvis ikke: */
    return erPalindrom(tekst.substring(1, tekst.length()-1));
  }

  public static void main(String[] args) {
    /* Følgende konstruksjon er lite brukt i boka, men vi instansierer
     * samme klasse som main() er i. Vi kunne også gjort metoden static.
     */
    Palindrom minPalindrom = new Palindrom();
    System.out.print("Er sommerferie et palindrom: "
                                + minPalindrom.erPalindrom("sommerferie") + "\n");
    System.out.print("Er Agnes i senga et palindrom: " +
                                minPalindrom.erPalindrom("Agnes i senga") + "\n");

  }
}
/* Utskrift:
Er sommerferie et palindrom: false
Er Agnes i senga et palindrom: true
*/
```

Kapittel 17.7

Oppgave 1

```
import java.util.*;

class BinaertSoekeTreTreeMap {
  private TreeMap<Integer, Integer> tre = new TreeMap<Integer, Integer>();

  public String toString() {
    String resultatTekst = "";
    TreeMap<Integer, Integer> hjelpeTre = new TreeMap<Integer, Integer>(tre);
    Integer tallUt = hjelpeTre.firstKey();
    while (tallUt != null) {
      resultatTekst = resultatTekst + tallUt.toString() + " ";
      try {
          hjelpeTre.remove(tallUt);
          tallUt = hjelpeTre.firstKey();
      }
      catch (NoSuchElementException e) {
          tallUt = null;
      }
    }
    return resultatTekst;
  }

  public void settInnVerdi(int verdi) {
    /* Både nøkkel og data er samme tall.*/
    tre.put(verdi, verdi);
```

```
  }

  /* Returnerer true dersom verdien finnes i treet. */
  public boolean søkEtterVerdi(int søkeVerdi) {
    /* Husk at equals() brukes, vi trenger vare et likt objekt å
     * søke med.
     */
    if (tre.get(søkeVerdi) != null) return true;
    return false;
  }
}
```

Også her kan vi bruke den samme testklienten.

Kapittel 18.3

Oppgave 1

Dette vil avhenge litt av nettleseren. I Netscape 4.7 vil appleten for eksempel avsluttes, lastes og startes igjen, det vil si at nettleseren kaller destroy() og så init() og start().

Kapittel 18.5

Oppgave 1

Slik blir programmet:

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

public class ParameterAddering extends JApplet {
  private Container innhold = getContentPane();
  private String tall1;
  private String tall2;

  public void init() {
    tall1 = getParameter("tall1");
    tall2 = getParameter("tall2");
    innhold.add(new Tegning(tall1, tall2));
  }
}

/*
 * Dette JPanel'et får beskjed om de parametrene som appleten har fått
 * gjennom konstruktøren.
 */
class Tegning extends JPanel {
  private String tall1, tall2;
  private int sum = 0;

  public Tegning(String appletensTall1, String appletensTall2) {
    tall1 = appletensTall1;
```

```
    tall2 = appletensTall2;
    sum = Integer.parseInt(tall1) + Integer.parseInt(tall2);
   }

   public void paintComponent(Graphics g) {
    super.paintComponent(g);
    g.drawString("Tall 1 som ble sendt inn i denne appleten er: " + tall1, 5, 50);
    g.drawString("Tall 2 som ble sendt inn i denne appleten er: " + tall2, 5, 65);
    g.drawString("Summen er: " + sum , 5, 80);
   }
}
```

Og vi kan bruke en *html*-fil som ser eksempelvis slik ut:

```
<html>
<head>
</head>
<body>

<h1>Applet som demonstrerer bruk av parametere</h1>

<applet code="ParameterAddering.class" width="500" height="100">
<param name="tall1" value="6">
<param name="tall2" value="7">

Nettleseren din støtter ikke Java-appleter, eller har det avskrudd.
</applet>
</body>
</html>
```