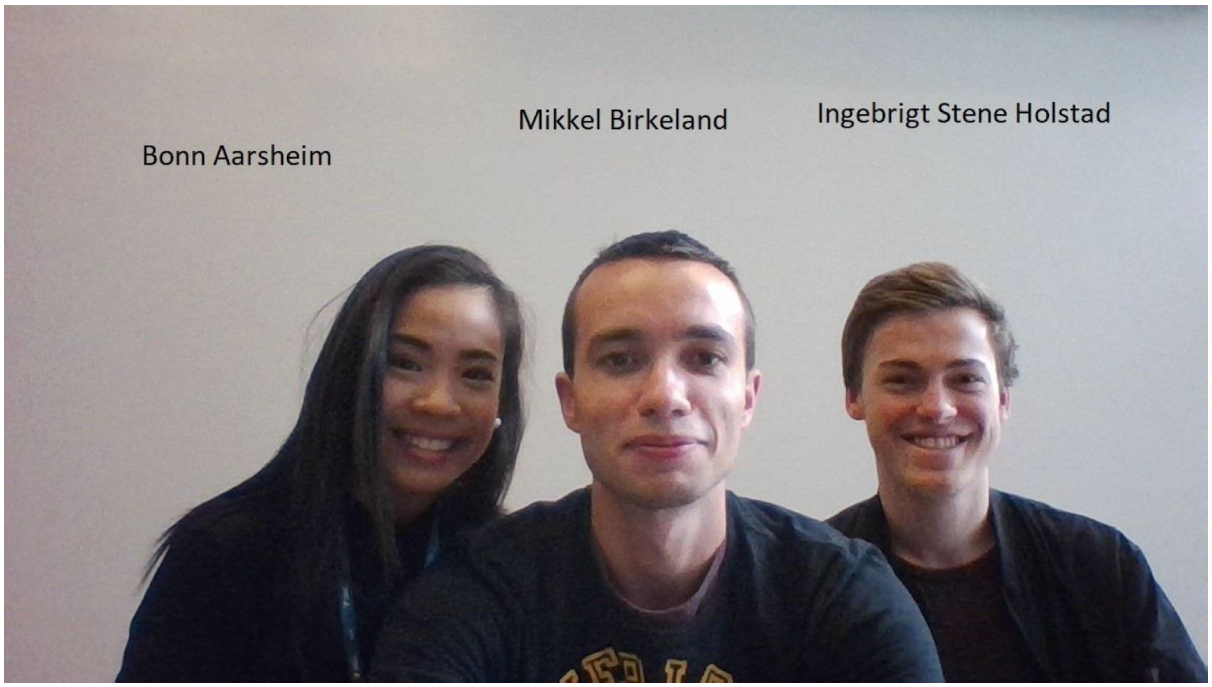
DAT102 ALGORITMER OG DATASTRUKTURER
Vår 2018 Oblig 2, øving 3 + øving 4



Oppgave 1

Kjøring av klientBingo med KjedetMengde:

```
6  public class KlientBingo {
7      // Oppretter 2 mengder med 75 bingokuler i hver.
8      // Tester om en spesiell bingokule er med i den ene mengden,
9      // og om de to mengdene er nøyaktig like.
10
11     public static void main(String[] a) {
12         final int ANTALL_BALLER = 75;
13
14         KjedetMengde<Bingokule> minMengde1 = new KjedetMengde<Bingokule>();
15         KjedetMengde<Bingokule> minMengde2 = new KjedetMengde<Bingokule>();
16         /*
17         TabellMengde<Bingokule> minMengde1 = new TabellMengde<Bingokule>();
18         TabellMengde<Bingokule> minMengde2 = new TabellMengde<Bingokule>();
19         */
20         Bingokule kule1 = null;
21         Bingokule kule2 = null;
```

```
Console ×
<terminated> KlientBingo [Java Application] C:\Program Files\Java\jre1.8.0_151\bin\javaw.exe (5 Mar 2018, 11:44:14

Antall kuler totalt: 75

kule 1 funnet i mengde 1
Like mengder
Ulike mengder
```

Kjøring av KlientBingo med TabellMengde:

```
6  public class KlientBingo {
7      // Oppretter 2 mengder med 75 bingokuler i hver.
8      // Tester om en spesiell bingokule er med i den ene mengden,
9      // og om de to mengdene er nøyaktig like.
10
11     public static void main(String[] a) {
12         final int ANTALL_BALLER = 75;
13         /*
14         KjedetMengde<Bingokule> minMengde1 = new KjedetMengde<Bingokule>();
15         KjedetMengde<Bingokule> minMengde2 = new KjedetMengde<Bingokule>();
16         */
17         TabellMengde<Bingokule> minMengde1 = new TabellMengde<Bingokule>();
18         TabellMengde<Bingokule> minMengde2 = new TabellMengde<Bingokule>();
19
20         Bingokule kule1 = null;
21         Bingokule kule2 = null;
```

```
Console ×
<terminated> KlientBingo [Java Application] C:\Program Files\Java\jre1.8.0_151\bin\javaw.exe (5 Mar 2018, 11:4

Antall kuler totalt: 75

kule 1 funnet i mengde 1
Like mengder
Ulike mengder
```

Klassen TabellMengde:

```
@Override
    public boolean equals(MengdeADT<T> m2) {
       boolean likeMengder = true;
       T element;

       // --- Oppgave ---
       Iterator<T> it = m2.oppramser();
       if(antall == m2.antall()) {
             while(it.hasNext() && likeMengder) {
                   if(!inneholder(it.next())) {
                         likeMengder = false;
                   }
             }
       }
       else {
             likeMengder = false;
       }
       return likeMengder;
    }
```

```java
@Override
    public T fjern(T element) {
       // Søker etter og fjerner element.Retur med null ved ikke-funn

       boolean funnet = false;
       T svar = null;
       // --- Oppgave ---
       if(erTom()) {
             System.out.println("Ikke noe å fjerne");
             return svar;
       }
       for (int i = 0; i < antall; i++) {
             if (tab[i].equals(element)) {
                   svar = tab[i];
                   tab[i] = tab[antall - 1];
                   tab[antall - 1] = null;
                   antall--;
             }
       }
       return svar;
    }
```

Klassen KjedetMengde:

```java
@Override
    public boolean equals(MengdeADT<T> m2) {
       boolean likeMengder = true;
       T element = null;
       // --- Oppgave ---
       Iterator<T> iterator = m2.oppramser();
       if(antall == m2.antall()) {
             while(iterator.hasNext() && likeMengder) {
                   if(!inneholder(iterator.next())) {
                         likeMengder = false;
                   }
             }
       }
       else {
             likeMengder = false;
       }
       return likeMengder;
    }
```

```java
@Override
    public T fjern(T element) {
       boolean funnet = false;
       LinearNode<T> forgjenger, aktuell;
       T resultat = null;
       // --- Oppgave ---
       if(!erTom()) {
             // Om du skal fjerne første node
             if(element.equals(start.getElement())) {
                   resultat = start.getElement();
                   start = start.getNeste();
                   funnet = true;
             }
             else { //Om du skal fjerne noe annet enn første node
                   forgjenger = start;
                   for (int nr = 2; nr < antall && !funnet; nr++) {
                         aktuell = forgjenger.getNeste();
                         if(aktuell.getElement().equals(element)) {
                               resultat = aktuell.getElement();
                               forgjenger.setNeste(aktuell.getNeste());
                               funnet = true;
                         }
                         forgjenger = forgjenger.getNeste();
                   }
             }

       }
       if(funnet) {
             antall--;
       }
       return resultat;
    }//
```

Oppgave 1 b)

Kjøring av Ordliste med KjedetMengde:

```java
package no.hib.dat102.mengde.klient;

import no.hib.dat102.mengde.adt.MengdeADT;

public class Ordliste {

    /**
     * @param args
     */
    public static void main(String[] args) {

        MengdeADT<String> ordListe1 = new KjedetMengde<String>(

        String[] ord = { "God", "dag", "Hans", "Hansen", "Hansa

        Scanner tastatur = new Scanner(System.in);
        // Legger til ordene i mengden ordListe1

        for (int i = 0; i < ord.length; i++) {
            ordListe1.leggTil(ord[i]);
        }
        MengdeADT<String> ordListe2 = new KjedetMengde<String>(

        System.out.print("Oppgi en streng, avslutt med zzz :");
        String streng = tastatur.nextLine();
        // Leser inn ord
        while (!streng.equals("zzz")) {

            if (ordListe1.inneholder(streng)) {
                System.out.println("ordListe1 inneholder " + st
            } else {
                System.out.println("ordListe1 inneholder ikke "
```

```
<terminated> Ordliste [Java Application] C:\Program Files\Ja
Oppgi en streng, avslutt med zzz :Mikkel
ordListe1 inneholder ikke Mikkel
Oppgi en streng, avslutt med zzz :Ola
ordListe1 inneholder ikke Ola
Oppgi en streng, avslutt med zzz :dag
ordListe1 inneholder dag
Oppgi en streng, avslutt med zzz :zzz
Utskrift av unionen av begge ordlistene
rute
God
Mikkel
Bergen
dag
Hansaby
buss
Ola
Ole
Hans
Hansen
Olsen
Utskrift av snittet av begge ordlistene
dag
Utskrift av differensen av begge ordlistene
Hansaby
Ole
God
buss
Olsen
Hansen
Hans
Bergen
rute
```

Kjøring av Ordliste med TabellMengde:

```
Oppgi en streng, avslutt med zzz :Mikkel
ordListe1 inneholder ikke Mikkel
Oppgi en streng, avslutt med zzz :Ola
ordListe1 inneholder ikke Ola
Oppgi en streng, avslutt med zzz :dag
ordListe1 inneholder dag
Oppgi en streng, avslutt med zzz :zzz
Utskrift av unionen av begge ordlistene
Ole
dag
Bergen
Ola
buss
Hans
Olsen
Mikkel
Hansen
Hansaby
God
rute
Utskrift av snittet av begge ordlistene
dag
Utskrift av differensen av begge ordlistene
Olsen
Ole
God
Hansen
rute
buss
Bergen
Hans
Hansaby
```

Klassen KjedetMengde:

```
@Override
    public MengdeADT<T> union(MengdeADT<T> m2) {
       MengdeADT<T> begge = new KjedetMengde<T>();
       LinearNode<T> aktuell = start;
       T element = null;
       /*
        * Oppgave
        */
       while(aktuell != null) {
             ((KjedetMengde<T>) begge).settInn(aktuell.getElement());
             aktuell = aktuell.getNeste();
       }
       Iterator<T> iterator = m2.oppramser();
       while(iterator.hasNext()) {
             element = iterator.next();
             if(!inneholder(element)){
                   ((KjedetMengde<T>)begge).settInn(element);
             }
       }
       return begge;
    }//

    @Override
    public MengdeADT<T> snitt(MengdeADT<T> m2) {
       MengdeADT<T> snittM = new KjedetMengde<T>();
       T element;
       /* Fyll ut...
        *
             if (this.inneholder(element))
                   ((KjedetMengde<T>) snittM).settInn(element);
       */
       Iterator<T> iterator = m2.oppramser();
       while(iterator.hasNext()) {
             element = iterator.next();
             if(this.inneholder(element)) {
                   ((KjedetMengde<T>) snittM).settInn(element);
             }
       }
       return snittM;
    }
```

```java
@Override
public MengdeADT<T> differens(MengdeADT<T> m2) {
   MengdeADT<T> differensM = new KjedetMengde<T>();
   T element;
   /*Fyll ut
    *
    */
   Iterator<T> iterator = this.oppramser();
   while(iterator.hasNext()) {
         element = iterator.next();
         if(!m2.inneholder(element)) {
               ((KjedetMengde<T>) differensM).settInn(element);
         }
   }
   return differensM;
}
```

```java
@Override
public boolean undermengde(MengdeADT<T> m2) {
   boolean erUnderMengde = true;
   //Fyll ut
   Iterator<T> iterator = m2.oppramser();
   if(this.antall < m2.antall()) {
         return false;
   }
   while(iterator.hasNext()) {
         T element = iterator.next();
         if(!this.inneholder(element)) {
               return false;
         }
   }
   return erUnderMengde;
}
```

Klassen TabellMengde:

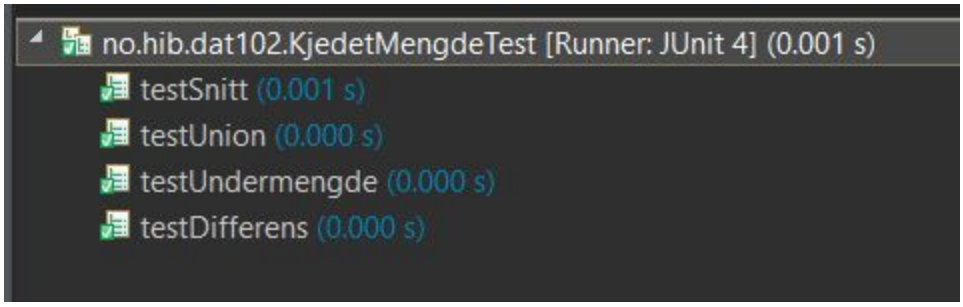
```
@Override
    public MengdeADT<T> union(MengdeADT<T> m2) {
       MengdeADT<T> begge = new TabellMengde<T>();
       T element = null;
       /*
        * Oppgave
        */
       for(int i = 0; i < antall; i++) {
             ((TabellMengde<T>) begge).settInn(tab[i]);
       }
       Iterator<T> iterator = m2.oppramser();
       while(iterator.hasNext()) {
             T ele = iterator.next();
             if(!inneholder(ele)) {
                   ((TabellMengde<T>) begge).settInn(ele);
             }
       }
       return begge;
    }//

    @Override
    public MengdeADT<T> snitt(MengdeADT<T> m2) {
       MengdeADT<T> snittM = new TabellMengde<T>();
       T element = null;
       /*
        * Fyll ut
        */
       Iterator<T> iterator = m2.oppramser();
       while(iterator.hasNext()) {
             element = iterator.next();
             if(this.inneholder(element)) {
                   ((TabellMengde<T>) snittM).settInn(element);
             }
       }
       return snittM;
    }

    @Override
    public MengdeADT<T> differens(MengdeADT<T> m2) {
       MengdeADT<T> differensM = new TabellMengde<T>();
```

```
        T element;
        /*
         * Fyll ut
         *
         * if (!m2.inneholder(element)) ((TabellMengde<T>)
differensM).settInn(element);
         */
        Iterator<T> iterator = this.oppramser();
        while(iterator.hasNext()) {
              T ele = iterator.next();
              if(!m2.inneholder(ele)) {
                    ((TabellMengde<T>) differensM).settInn(ele);
              }
        }

        return differensM;
    }
```

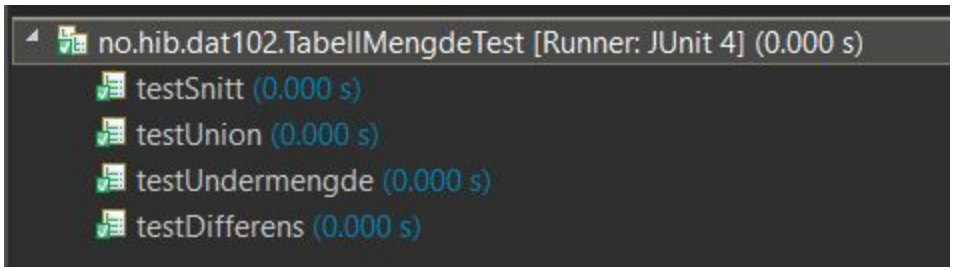
```
@Override
    public boolean undermengde(MengdeADT<T> m2) {
        boolean erUnderMengde = true;
        // Fyll ut
        Iterator<T> iterator = m2.oppramser();
        if(this.antall < m2.antall()) {
              return false;
        }
        while(iterator.hasNext()) {
              T ele = iterator.next();
              if(!this.inneholder(ele)) {
                    return false;
              }
        }
        return erUnderMengde;
    }
```

Oppgave 1 c)

Testresultater for KjedetMengde:



Testresultater TabellMengde:



Abstrakt testklasse:

```java
package no.hib.dat102;

import static org.junit.Assert.*;

import org.junit.Before;
import org.junit.Test;

import no.hib.dat102.mengde.adt.MengdeADT;

public abstract class AbstractTest {
    private MengdeADT<Integer> intListe1;
    private MengdeADT<Integer> intListe2;
    private MengdeADT<Integer> svar;
    private MengdeADT<Integer> fasit;


    abstract MengdeADT<Integer> reset();

    @Before
    public void setup() {
```

```
        intListe1 = reset();
        intListe2 = reset();
        svar = reset();
        fasit = reset();

        for(int i = 1; i < 6; i++) {
              intListe1.leggTil(i*2);
              intListe2.leggTil((int)Math.pow(i, 2));
        }
    }

    @Test
    public void testUnion() {

        svar = intListe1.union(intListe2);

        for(int i = 1; i < 6; i++) {
              fasit.leggTil(i*2);
              fasit.leggTil((int)Math.pow(i, 2));
        }

        assertTrue(svar.equals(fasit));
        svar.fjernTilfeldig();
        assertFalse(svar.equals(fasit));
    }

    @Test
    public void testSnitt() {
        svar = intListe1.snitt(intListe2);
        fasit.leggTil(4);

        assertTrue(svar.equals(fasit));
        svar.fjernTilfeldig();
        assertFalse(svar.equals(fasit));
    }

    @Test
    public void testDifferens() {
        svar = intListe1.differens(intListe2);
        fasit.leggTil(2);
        fasit.leggTil(6);
        fasit.leggTil(8);
```

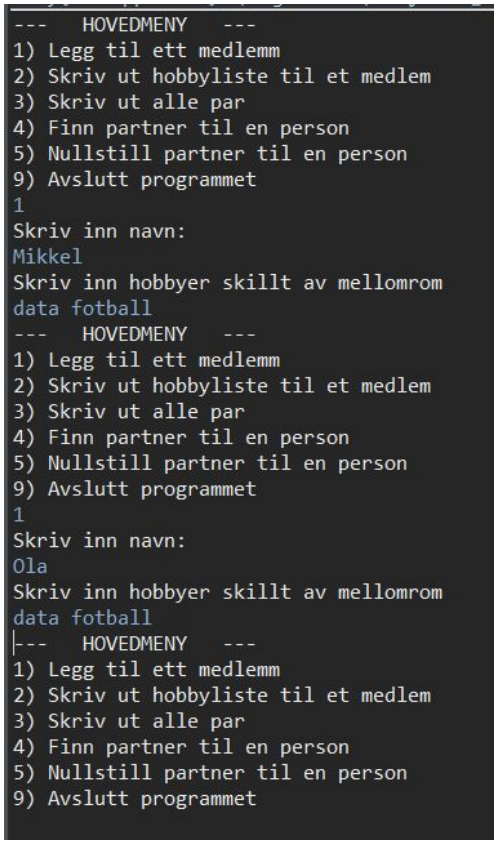
```
        fasit.leggTil(10);

        assertTrue(svar.equals(fasit));
        svar.leggTil(101);
        assertFalse(svar.equals(fasit));
    }

    @Test
    public void testUndermengde() {

        for(int i = 1; i < 6; i++) {
              intListe2.leggTil(i*2);
              intListe2.leggTil((int)Math.pow(i, 2));
        }

        assertTrue(intListe2.undermengde(intListe1));

        intListe1.leggTil(100);

        assertFalse(intListe2.undermengde(intListe1));
    }

}
```

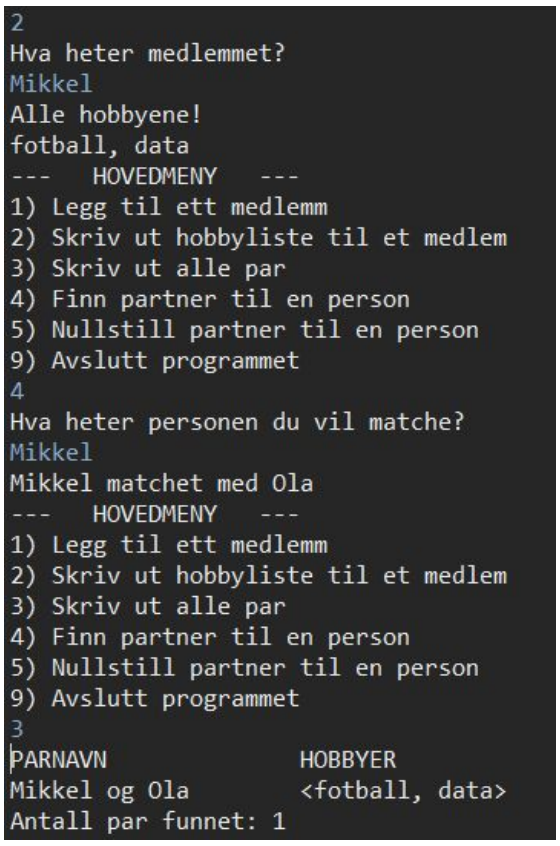
```
public class TabellMengdeTest extends AbstractTest{
    public TabellMengde<Integer> reset() {
        return new TabellMengde<Integer>();
    }
}
```

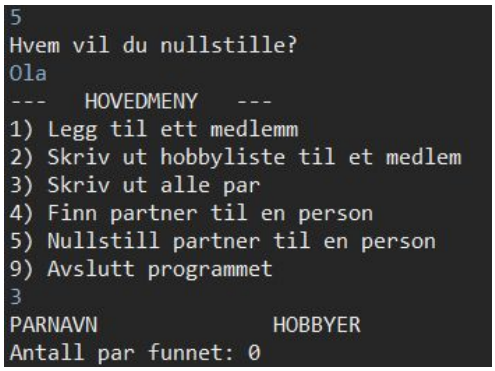
```
public class KjedetMengdeTest extends AbstractTest {
    public KjedetMengde<Integer> reset() {
        return new KjedetMengde<Integer>();
    }
}
```

Oppgave 2

Viser hovedmeny og “legg til” funksjonen, deretter matches medlemmer og alle par skrives ut. Siste bildet demonstrerer nullstilling av match på et par.







Klassen Datakontakt:

```
package no.hib.dat102.oppgave2;

public class Datakontakt {
    private int antallMedlemmer;
    private Medlem[] medlemstabell;

    private final static int STDK = 100;

    public Datakontakt(int storrelse) {
       antallMedlemmer = 0;
       medlemstabell = new Medlem[storrelse];
    }

    public Datakontakt() {
       this(STDK);
    }

    public int getAntallMedlemmer() {
       return antallMedlemmer;
    }

    public void setAntallMedlemmer(int antallMedlemmer) {
       this.antallMedlemmer = antallMedlemmer;
    }

    public Medlem[] getMedlemstabell() {
       return medlemstabell;
    }

    public void setMedlemstabell(Medlem[] medlemstabell) {
       this.medlemstabell = medlemstabell;
    }

    public void leggTilMedlem(Medlem medlem){
       if(antallMedlemmer == medlemstabell.length) {
             utvid();
       }
       medlemstabell[antallMedlemmer] = medlem;
       antallMedlemmer++;
```

```
    }

    public void utvid() {
       Medlem[] utvidetTab = new Medlem[medlemstabell.length * 2];
       System.arraycopy(medlemstabell, 0, utvidetTab, 0, antallMedlemmer);
       medlemstabell = utvidetTab;
    }

    public int finnMedlemsIndex(String medlemsnavn) {
       for(int i = 0; i < antallMedlemmer; i++) {
             if(medlemstabell[i].getNavn().equals(medlemsnavn)) {
                   return i;
             }
       }
       return -1;
    }

    public int finnPartnerFor(String medlemsnavn) {
       int medlemSok = finnMedlemsIndex(medlemsnavn);
       Medlem mSok = medlemstabell[medlemSok];

       for(int i = 0; i < antallMedlemmer; i++) {
             if(i != medlemSok && mSok.passerTil(medlemstabell[i])) {
                   mSok.setStatusIndex(i);
                   medlemstabell[i].setStatusIndex(medlemSok);
                   return i;
             }
       }
       return -1;
    }

    public void tilbakestillStausIndeks(String medlemsnavn) {
       int medlemSok = finnMedlemsIndex(medlemsnavn);
       int partnerIndex = medlemstabell[medlemSok].getStatusIndex();

       if(partnerIndex != -1) {
             medlemstabell[medlemSok].setStatusIndex(-1);
             medlemstabell[partnerIndex].setStatusIndex(-1);
       }
    }
}
```

Klassen Hobby:

```
package no.hib.dat102.oppgave2;

public class Hobby {
    private String hobbyNavn;

    public Hobby(String hobby) {
       hobbyNavn = hobby;
    }

    public String getHobbyNavn() {
       return hobbyNavn;
    }

    public void setHobbyNavn(String hobbyNavn) {
       this.hobbyNavn = hobbyNavn;
    }

    public String toString() {
       return hobbyNavn;
    }

    public boolean equals(Object hobby2) {
       Hobby hobbyDenAndre = (Hobby) hobby2;

       return (hobbyNavn.equals(hobbyDenAndre.getHobbyNavn()));
    }
}
```

Klassen Medlem:

```
package no.hib.dat102.oppgave2;

import no.hib.dat102.mengde.kjedet.KjedetMengde;

public class Medlem {
    private String navn;
    private KjedetMengde<Hobby> hobbyer;
    private int statusIndex;

    public Medlem(String navn, KjedetMengde<Hobby> hobbyer) {
```

```
        this.navn = navn;
        this.hobbyer = hobbyer;
        statusIndex = -1;
    }

    public Medlem() {
        this("Ukjent", new KjedetMengde<Hobby>());
    }

    public String getNavn() {
        return navn;
    }

    public void setNavn(String navn) {
        this.navn = navn;
    }

    public KjedetMengde<Hobby> getHobbyer() {
        return hobbyer;
    }

    public void setHobbyer(KjedetMengde<Hobby> hobbyer) {
        this.hobbyer = hobbyer;
    }

    public int getStatusIndex() {
        return statusIndex;
    }

    public void setStatusIndex(int statusIndex) {
        this.statusIndex = statusIndex;
    }

    public boolean passerTil(Medlem medlem2) {
        return statusIndex == -1 ? hobbyer.equals(medlem2.getHobbyer()):
false;
    }
}
```

Klassen Meny, denne inneholder også main-metoden:

```
package no.hib.dat102.oppgave2;

import java.util.Scanner;

public class Meny {
    public static void Meny(Datakontakt arkiv) {
       boolean run = true;
       Scanner sc = new Scanner(System.in);

       while(run) {
             System.out.println("---   HOVEDMENY   ---\t");
             System.out.println("1) Legg til ett medlemm");
             System.out.println("2) Skriv ut hobbyliste til et medlem");
             System.out.println("3) Skriv ut alle par");
             System.out.println("4) Finn partner til en person");
             System.out.println("5) Nullstill partner til en person");
             System.out.println("9) Avslutt programmet");
             int valg = sc.nextInt();
             sc.nextLine();

             if(valg == 1) {
                   Medlem medlem = Tekstgrensesnitt.lesMedlem();
                   arkiv.leggTilMedlem(medlem);
             }
             else if(valg == 2) {
                   System.out.println("Hva heter medlemmet?");
                   String navn = sc.nextLine();
                   int pos = arkiv.finnMedlemsIndex(navn);
                   Medlem[] liste = arkiv.getMedlemstabell();
                   Tekstgrensesnitt.skrivHobbyListe(liste[pos]);
             }
             else if(valg == 3) {
                   Tekstgrensesnitt.skrivParListe(arkiv);
             }
             else if(valg == 4) {
                   System.out.println("Hva heter personen du vil matche?");
                   String navn = sc.nextLine();
                   int pos = arkiv.finnPartnerFor(navn);
                   if(pos == -1) {
                         System.out.println(navn + " matchet ikke med
noen");
```

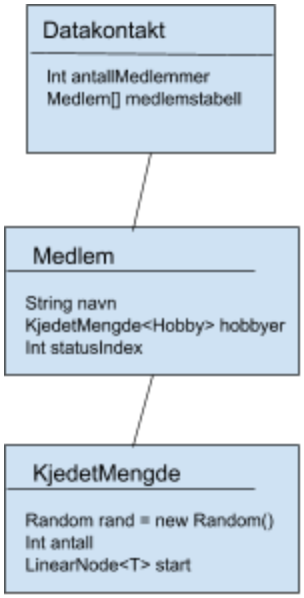
```
                }
                else {
                        Medlem[] liste = arkiv.getMedlemstabell();
                        System.out.println(navn + " matchet med " +
liste[pos].getNavn());
                }
            }
            else if(valg == 5) {
                System.out.println("Hvem vil du nullstille?");
                String navn = sc.nextLine();
                arkiv.tilbakestillStausIndeks(navn);
            }
            else if(valg == 9) {
                run = false;
                System.out.println("Avslutter programmet");
            }
        }
    }

    public static void main(String[] args) {
        Datakontakt arkiv = new Datakontakt();
        Meny(arkiv);
    }
}
```

Klassen Tekstgrensesnitt:

```
package no.hib.dat102.oppgave2;

import java.util.Scanner;

import no.hib.dat102.mengde.kjedet.KjedetMengde;

public class Tekstgrensesnitt {
    public static Medlem lesMedlem() {
        Scanner sc = new Scanner(System.in);
        System.out.println("Skriv inn navn:");
        String navn = sc.nextLine();
        System.out.println("Skriv inn hobbyer skillt av mellomrom");
        String hobbyerString = sc.nextLine();
```

```
        String[] hobbyListe = hobbyerString.split(" ");
        KjedetMengde<Hobby> mengde = new KjedetMengde<>();

        for(String s : hobbyListe) {
              mengde.leggTil(new Hobby(s));
        }

        return new Medlem(navn, mengde);
    }

    public static void skrivHobbyListe(Medlem medlem) {
        System.out.println("Alle hobbyene!");
        System.out.println(medlem.getHobbyer().toString());
    }

    public static void skrivParListe (Datakontakt arkiv) {
        Medlem[] listen = arkiv.getMedlemstabell();
        int antallPar = 0;

        System.out.format("%-20s %s%n", "PARNAVN", "HOBBYER");
        for(int i = 0; i < arkiv.getAntallMedlemmer(); i++) {
              if(listen[i].getStatusIndex() != -1 &&
listen[i].getStatusIndex() > i) {
                    int partnerIndex = listen[i].getStatusIndex();
                    System.out.format("%-20s", listen[i].getNavn() + " og "
+ listen[partnerIndex].getNavn());
                    System.out.format(" %s%n", "<" +
listen[i].getHobbyer().toString() + ">");
                    antallPar++;
              }
        }
        System.out.println("Antall par funnet: " + antallPar);
    }
}
```

Objektdiagram som viser hvordan Datakontakt og Medlem er implementert:



Oppgave 3

a) Abstrakt datatype (ADT) er en datatype som sier hvilken operasjoner som skal bli utført, men ikke hvordan de blir implementert. Den spesifiserer ikke hvordan data blir organisert i minnet og ikke hvordan algoritmen skal se ut.

   Datastruktur er en måte å organisere et datasett

b) En algoritme er operasjoner som skal utføres for å løse et problem. Altså en nøyaktig og fullstendig beskrivelse av fremgangsmåten for å løse problemet.

c) Generisk type er en typeparameter som spesifiserer hvilken datatype en samling objekter kan inneholde. Hensikten er å skrive kode som kan bli brukt om igjen for mange forskjellige type objekter. Genereisk datatype tillater oss å bruke objektene og metodene på alle datatyper senere, slik at det blir godt gjennbrukspotensiale.

   Ved å bruke generisk datatype på samlinger kan feil bli oppdaget ved kompilering, fordi kompilatoren sjekker alle operasjonene som tilfører objekter til samlingen har riktig datatype. I tillegg vil typekonvertering være unødvendig, fordi typen til objekter som hentes ut fra samlingen allerede er kjent av kompilatoren.

d) ADT stabel: I en stabel kan du legge elementer på i en ende, og fjerne fra den samme enden, akkurat som med en stabel tallerkener. Stakken er altså en liste med LIFO-struktur: Last In, First Out. Å legge noe til i stabelen blir gjerne kallet "push", og fjerning "pop".

ADT kø: En kø følger first-in-first-out (FIFO) prinsippet. Elementer kan innsettes når som helst, men bare elementet som har befunnet seg lengst i køen kan bli tatt ut. Elementet settes inn bakerst, og fjernes fra fronten av køen.

e) I A) blir operasjonen utført 1000*n ganger. I stor-O notasjon blir dette n ganger. I B) blir operasjonen utført n^2 ganger. Så n^2 i stor-O notasjon. Dermed er A) mest tidseffektiv i følge stor-O notasjon.

For alle verdier n <1000, vil B) være raskere enn A).

Oppgave 4

a) Enhetstesting er en test hvor de individuelle enhetene i koden blir testet uten påvirkning av andre deler. Hensikten med en enhetstest er å verifisere at koden utfører det som var forventet.
@Test er en notasjon som spesifiserer at metoden er en testmetode.
@Before er en notasjon blir skrevet før en kode, som vi vil skal utføres før testmetodene i klassen blir utført. Metoden med denne "taggen" vil bli utført før hver eneste metode med "test-taggen".

b) KjedetOrdnetListe

```
public void leggTil(T element) {
	//...Fyll ut
	LinearNode<T> aktuell = foerste;
	LinearNode<T> forige = null;
	LinearNode<T> nyNode = new LinearNode<T>(element);
	boolean lagtTil = false;

	//Om kjeden er helt tom fra før
	if(foerste == null) {
		foerste = nyNode;
		siste = nyNode;
		antall++;
		lagtTil = true;
	}

	while(aktuell != null && !lagtTil) {
		//Om den nye noden skal inn på første plass i kjeden
		if(element.compareTo(foerste.getElement()) <= 0) {
			foerste = nyNode;
			nyNode.setNeste(aktuell);
			antall++;
			lagtTil = true;
```

```
            }
            //Om den nye noden skal legges til i slutten av kjeden
            else if(element.compareTo(siste.getElement()) > 0) {
                  siste.setNeste(nyNode);
                  siste = nyNode;
                  antall++;
                  lagtTil = true;
            }
            //Om den nye noden skal inn på noe annet enn første plass
            else if(element.compareTo(aktuell.getElement()) <= 0) {
                  forige.setNeste(nyNode);
                  nyNode.setNeste(aktuell);
                  antall++;
                  lagtTil = true;
            }
            forige = aktuell;
            aktuell = aktuell.getNeste();
      }
   }
```

TabellOrdnetListe

```
@Override
   public void leggTil(T element) {
      //...Fyll ut
      //Sjekker om listen er full, og utvider om det trengs
      if(bak == liste.length) {
            utvid();
      }
      int i = 0;
      //Finner hvor elementet skal
      while(i < bak && element.compareTo(liste[i]) > 0) {
            i++;
      }
      int j = bak;
      while(j > i) {
            liste[j] = liste[j-1];
            j--;
      }
      liste[i] = element;
      bak++;
   }
```

KjedetOrdnetListe

```java
@Override
    public T fjernFoerste() {
       T svar = null;
       //...Fyll ut
       if(!erTom()) {
             svar = foerste.getElement();
             foerste = foerste.getNeste();
             antall--;
       }
       if(erTom()) {
             siste = null;
       }
       return svar;
    }

    @Override
    public T fjernSiste() {
       T svar = null;
       //...Fyll ut
       if(!erTom()) {
             svar = siste.getElement();
             antall--;
             if(antall == 0) {
                   foerste = null;
                   siste = null;
             }else {
                   siste = foerste;
                   for(int i = 1; i < antall; i++) {
                         siste = siste.getNeste();
                   }
                   siste.setNeste(null);
             }
       }
       return svar;
    }
```

TabellOrdnetListe

```
@Override
    public T fjernSiste() {
       T resultat = null;
       //... Fyll ut
       if(!erTom()) {
             resultat = liste[bak-1];
             liste[bak-1] = null;
             bak--;
       }
       return resultat;
    }

    @Override
    public T fjernFoerste() {
       T resultat = null;
       //... Fyll ut
       if(!erTom()) {
             resultat = liste[0];
             bak--;
             //Flytter alle elementer ett hakk nedover i listen
             for(int i = 0; i < bak; i++) {
                   liste[i] = liste[i + 1];
             }
       }
       return resultat;
    }
```

c) finn-metode og fjern-metode i TabellOrdnetListe

```
@Override
    public T fjern(T element) {
       // ...Fyll ut
       int index = finn(element);

       if(index != -1) {
             bak--;
             //Flytter alle elementer ett hakk nedover i listen
             for(int i = index; i < bak; i++) {
                   liste[i] = liste[i + 1];
             }
       }
```

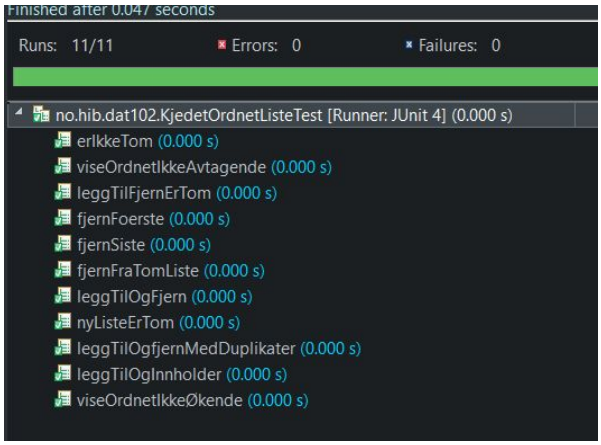
```
        else {
              element = null;
        }
        return element;

    }

    private int finn(T el) {
        int i = 0, resultat = IKKE_FUNNET;
        //...Fyll ut
        if(erTom()) {
              return resultat;
        }
        while(resultat == -1 && i < bak) {
              if(liste[i].equals(el)) {
                    resultat = i;
              }
              else {
                    i++;
              }
        }
        if(resultat == -1) {
              System.out.println("Fant ikke elementet");
        }
        return resultat;
    }
```

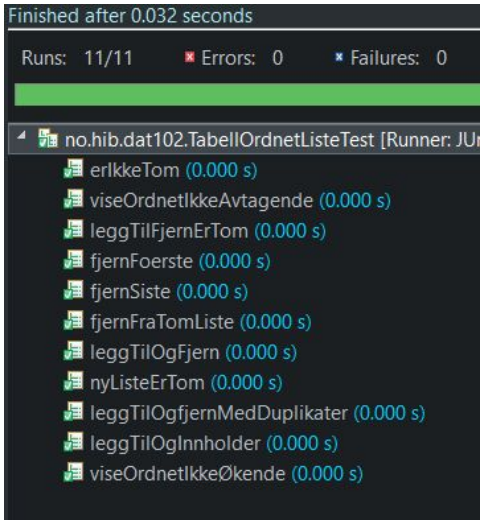
KjedetOrdnetListe

```
@Override
    public T fjern(T element) {
        T svar = null;
        LinearNode<T> forrige = null, denne = foerste;
        while (denne != null && element.compareTo(denne.getElement()) > 0) {
              forrige = denne;
              denne = denne.getNeste();
        }
        if (denne != null && element.equals(denne.getElement())) { // funnet
              antall--;
              svar = denne.getElement();
              if (forrige == null) {       // Første element
                    foerste = foerste.getNeste();
```

```
                if (foerste == null) { // Tom liste
                        siste = null;
                }
            } else { // Inni listen eller bak
                forrige.setNeste(denne.getNeste());
                if (denne == siste) { // bak
                        siste = forrige;
                }
            }
        } // ikke-funn
        return svar;
    }
```

d)






```
/**
     * Tester ordning ikke-avtagende
     *
     */
    @Test
    public final void viseOrdnetIkkeAvtagende() {
        //... Fyll ut
        // ... Legg til elementer og bruk fjernFoerste
        liste.leggTil(e1);
        liste.leggTil(e2);
        liste.leggTil(e5);
        liste.leggTil(e0);
        liste.leggTil(e4);
```

```
        liste.leggTil(e3);
        assertEquals(e0, liste.fjernFoerste());
        assertEquals(e1, liste.fjernFoerste());
        assertEquals(e2, liste.fjernFoerste());
        assertEquals(e3, liste.fjernFoerste());
        assertEquals(e4, liste.fjernFoerste());
        assertEquals(e5, liste.fjernFoerste());

    }
```

```
/**
     * Tester leggTil og fjern med like verdier.
     */
    @Test
    public final void leggTilOgfjernMedDuplikater() {
        //... Fyll ut med å legge til passende elementer
        liste.leggTil(e0);
        liste.leggTil(e1);
        liste.leggTil(e4);
        liste.leggTil(e1);
        liste.leggTil(e2);
        liste.leggTil(e3);
        assertEquals(e0, liste.fjern(e0));
        assertEquals(e1, liste.fjern(e1));
        assertEquals(e4, liste.fjern(e4));
        assertEquals(e1, liste.fjern(e1));
        assertEquals(e2, liste.fjern(e2));
        assertEquals(e3, liste.fjern(e3));

    }
```

```java
/**
     * Tester om leggTil-fjern på en tom liste gir en tom liste.
     */
    @Test
    public final void leggTilFjernErTom() {
       //...Fyll ut. Legg inn elementer og fjern de
       liste.leggTil(e1);
       liste.leggTil(e2);
       liste.fjern(e1);
       liste.fjern(e2);
       assertTrue(liste.erTom());
    }
```

```java
package no.hib.dat102;


import no.hib.dat102.adt.OrdnetListeADT;
import no.hib.dat102.tabell.TabellOrdnetListe;

public class TabellOrdnetListeTest extends ListeADTTest{

    public OrdnetListeADT<Integer> reset() {
       return new TabellOrdnetListe<Integer>();
    }

}
```

```java
package no.hib.dat102;

import no.hib.dat102.adt.OrdnetListeADT;
import no.hib.dat102.kjedet.KjedetOrdnetListe;

public class KjedetOrdnetListeTest extends ListeADTTest{

    public OrdnetListeADT<Integer> reset() {
       return new KjedetOrdnetListe<Integer>();
    }

}
```

Oppgave 5)

```java
package no.hib.dat102;

public class Person implements Comparable<Person>{
    private String fornavn;
    private String etternavn;
    private int fodtaar;

    public Person(String fornavn, String etternavn, int fodtaar) {
       this.fornavn = fornavn;
       this.etternavn = etternavn;
       this.fodtaar = fodtaar;
    }

    public Person() {
       this("Ukjent", "Ukjent", 0000);
    }

    public String getFornavn() {
       return fornavn;
    }

    public void setFornavn(String fornavn) {
       this.fornavn = fornavn;
    }

    public String getEtternavn() {
       return etternavn;
    }

    public void setEtternavn(String etternavn) {
       this.etternavn = etternavn;
    }

    public int getFodtaar() {
       return fodtaar;
    }

    public void setFodtaar(int fodtaar) {
       this.fodtaar = fodtaar;
    }
```

```
    @Override
    public String toString() {
       return fodtaar + " " + etternavn + ", " + fornavn;
    }

    public int compareTo(Person denAndre) {
       if(fodtaar < denAndre.getFodtaar()) {
             return -1;
       }
       else if(fodtaar > denAndre.getFodtaar()) {
             return 1;
       }
       else if(etternavn.compareTo(denAndre.getEtternavn()) != 0) {
             return etternavn.compareTo(denAndre.getEtternavn());
       }
       else if(fornavn.compareTo(denAndre.getFornavn()) != 0) {
             return fornavn.compareTo(denAndre.getFornavn());
       }
       else {
             return 0;
       }
    }


}
```

```
package no.hib.dat102.klient;

import java.util.Scanner;

import no.hib.dat102.Person;
import no.hib.dat102.SirkulaerKoe;
import no.hib.dat102.kjedet.KjedetOrdnetListe;
import no.hib.dat102.tabell.*;

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

            SirkulaerKoe<Person> sirKoe = new SirkulaerKoe<>();
            TabellOrdnetListe<Person> tabOrdListe = new
```

```
TabellOrdnetListe<>();
            KjedetOrdnetListe<Person> kjedetOrdListe = new
KjedetOrdnetListe<>();


            for(int i = 0; i < 4; i++) {
                  Person p = lagPerson();
                  sirKoe.innKoe(p);
                  tabOrdListe.leggTil(p);
                  kjedetOrdListe.leggTil(p);
            }
            //Tester sirkulær kø
            System.out.println("\nKjøring av sirkulær kø\n");

            while(sirKoe.antall() != 0) {
                  System.out.println("Tar person ut av køen");
                  System.out.println(sirKoe.utKoe());
            }

            //Tester orndet liste
            System.out.println("\nKjøring av ordnet liste i stigende
alder\n");
            int antall = tabOrdListe.antall();
            for(int i = 0; i < antall; i++) {
                  System.out.println("Tar person ut av listen");
                  System.out.println(tabOrdListe.fjernSiste());
            }

            //Tester kjedet ordnet liste
            System.out.println("\nKjøring av kjedet ordnet liste i stigende
alder\n");
            int antallKjede = kjedetOrdListe.antall();
            for(int i = 0; i < antallKjede; i++) {
                  System.out.println("Tar person ut av listen");
                  System.out.println(kjedetOrdListe.fjernSiste());
            }


      }

      public static Person lagPerson() {
            Scanner sc = new Scanner(System.in);
```

```
            System.out.println("Skriv inn fødselsdato:");
            int fDato = sc.nextInt();
            sc.nextLine();
            System.out.println("Skriv inn etternavn:");
            String etternavn = sc.nextLine();
            System.out.println("Skriv inn fornavn:");
            String fornavn = sc.nextLine();
            return new Person(fornavn, etternavn, fDato);
      }
}
```

Kjøring:

```
Skriv inn fødselsdato:
1993
Skriv inn etternavn:
Hansen
Skriv inn fornavn:
Ola
Skriv inn fødselsdato:
2000
Skriv inn etternavn:
Nordman
Skriv inn fornavn:
Kari
Skriv inn fødselsdato:
1996
Skriv inn etternavn:
Nordman
Skriv inn fornavn:
Ola
Skriv inn fødselsdato:
1991
Skriv inn etternavn:
Johansen
Skriv inn fornavn:
Erik
```

```
Kjøring av sirkulær kø

Tar person ut av køen
1993 Hansen, Ola
Tar person ut av køen
2000 Nordman, Kari
Tar person ut av køen
1996 Nordman, Ola
Tar person ut av køen
1991 Johansen, Erik

Kjøring av ordnet liste i stigende alder

Tar person ut av listen
2000 Nordman, Kari
Tar person ut av listen
1996 Nordman, Ola
Tar person ut av listen
1993 Hansen, Ola
Tar person ut av listen
1991 Johansen, Erik

Kjøring av kjedet ordnet liste i stigende alder

Tar person ut av listen
2000 Nordman, Kari
Tar person ut av listen
1996 Nordman, Ola
Tar person ut av listen
1993 Hansen, Ola
Tar person ut av listen
1991 Johansen, Erik
```

Oppgave 6

Klassen Balansering

```
package no.hvl.dat102;

import java.io.*;

import no.hib.dat102.kjedet.KjedetStabel;

public class Balansering {
    // Her opphever du kommentarsetning når du har fått lagt inn
    // nødvendig kode
    // SirkulaerStabel<Parentesinfo>stabel = new
    // SirkulaerStabel<Parentesinfo>();
    KjedetStabel<Parentesinfo> stabel = new KjedetStabel<>();

    private boolean passer(char åpent, char lukket) {
       switch (åpent) {
       case '(':
             return (lukket == ')');
       case '[':
             return (lukket == ']');
       case '{':
             return (lukket == '}');
       default:
             return false;
       }
    }//

    public void foretaBalansering(String innDataStreng, int linjenr) {

       int lengde = innDataStreng.length();
       // Fyll ut
       for(int i = 0; i < lengde; i++) {
             char c = innDataStreng.charAt(i);
             if(c == '(' || c == '[' || c == '{') {
                   stabel.push(new Parentesinfo(linjenr, i, c));
             }
             if(c == ')' || c == ']' || c == '}') {
                   //Om stabelen er tom
                   if(stabel.erTom()) {
```

```
                        System.out.println("Traff på lukkesymbol " + c + "
uten å finne åpnesymbol");
                  }
                  //Om de ikke matcher
                  else{
                        char venstre = stabel.pop().hentVenstreparentes();
                        if(!passer(venstre, c)) {
                              System.out.println("Fant lukkesymbol " + c +
", men det matchet ikke " + venstre);
                        }
                  }
            }
      }


   }//

   public void lesFraFil(String filnavn) {
      System.out.println("Begynner innlesning av fil");
      FileReader tekstFilLeser = null;
      try {
            tekstFilLeser = new FileReader(filnavn);
      } catch (FileNotFoundException unntak) {
            System.out.println("Finner ikke filen!");
            System.exit(-1);
      }

      BufferedReader tekstLeser = new BufferedReader(tekstFilLeser);
      String linje = null;
      int linjenr = 0;
      try {
            linje = tekstLeser.readLine();
            while (linje != null) {
                  // kalle metode her!
                  // Fyll ut
                  foretaBalansering(linje, linjenr);
                  linje = tekstLeser.readLine();
            } // while
            //Stabelen er ikke tom etter å ha gått gjennom filen
            if(!stabel.erTom()) {
                  System.out.println("Stabelen er ikke tom, så det mangler
lukketegn i filen");
```

```
                    while(!stabel.erTom()) {

System.out.println(stabel.pop().hentVenstreparentes() + " var igjen i
stabelen");
                    }
              }
       }

       catch (IOException unntak) {
              System.out.println("Feil ved innlesing!");
              System.exit(-1);
       }
       try {
              tekstFilLeser.close();
       } catch (IOException unntak) {
              System.out.println("Feil ved lukking av fil!");
       }

    }// metode

}// class
```

Kjøring av denne java-filen:

```
public class KlientBalansering{
      public static void main(String[] args{
      final String filnavn = "data.txt";
      //Leser inn en tekst fra fil
      Balansering balansering = new Balansering();
      balansering.lesFraFilfilnavn);
 }//main

}//class
```

```
Begynner innlesning av fil
Fant lukkesymbol ), men det matchet ikke {
Fant lukkesymbol }, men det matchet ikke (
```

Kjøring ved innlesning av denne filen (Data2.txt)

```
package no.hvl.dat102;
public class KlientBalansering{
      public static void main(String[] args){
      final String filnavn = "Data.txt";
      //Leser inn en tekst fra fil
      Balansering balansering = new Balansering();
      balansering.lesFraFil(filnavn);
 //main

}//class
```

```
Begynner innlesning av fil
Stabelen er ikke tom, så det mangler lukketegn i filen
{ var igjen i stabelen
```

Kjøring ved innlesning av Data3.txt, denne filen skal ikke ha noen parantesfeil.

```
package no.hvl.dat102;
public class KlientBalansering{
      public static void main(String[] args){
      final String filnavn = "Data2.txt";
      //Leser inn en tekst fra fil
      Balansering balansering = new Balansering();
      balansering.lesFraFil(filnavn);
 }//main

}//class
```

```
Begynner innlesning av fil
```