Eksamen i IN1010 og INF1010

Stein Gjessing Dag Langmyhr

4. juni 2018

Generelle kommentarer

- Dette er et løsnings*forslag* og ikke en fasit. Det er bare én av mange ulike måter å løse oppgaven på.
- I dette oppgavesettet er noen spesifikasjoner utelatt, for eksempel datastrukturen for Flygning (oppgave 5) eller hvordan Seterad skal lagre Sete-ene. Dette er helt bevisst, og dere står da fritt til hvordan dere vil implementere dette. Alle fornuftige løsninger vil bli godtatt.
- I koden her er det lagt inn noen utskrifter for å teste koden; det er ikke forventet at dere skriver slikt på eksamen.

Oppgave 1: Leting i tekst

```
class Stringhjelper {
    static int inneholder(String s, String t) {
        int sLen = s.length(), tLen = t.length();

        for (int is = 0;  is <= sLen-tLen;  is++) {
            boolean eq = true;
            for (int it = 0;  it < tLen;  it++) {
                if (s.charAt(is+it) != t.charAt(it)) {
                    eq = false;  break;
                }
            }
            if (eq) return is;
        }
        return -1;
    }
}
```

Oppgave 2: Prioritetskø

```
class Prioritetsko <T> {
    private class Node {
        T data;
        int prioritet;
        Node neste = null;

        Node (T inn, int prio) {
            data = inn;  prioritet = prio;
        }
    }

    private Node forste = null;
    private int ant = 0;   // Antall noder i listen (>= 0)

    public void settInn(T inn, int prio) {
        Node ny = new Node(inn, prio);   ant++;

        if (forste == null)
            forste = ny;
        else if (ny.prioritet <= forste.prioritet) {
            ny.neste = forste;  forste = ny;
```

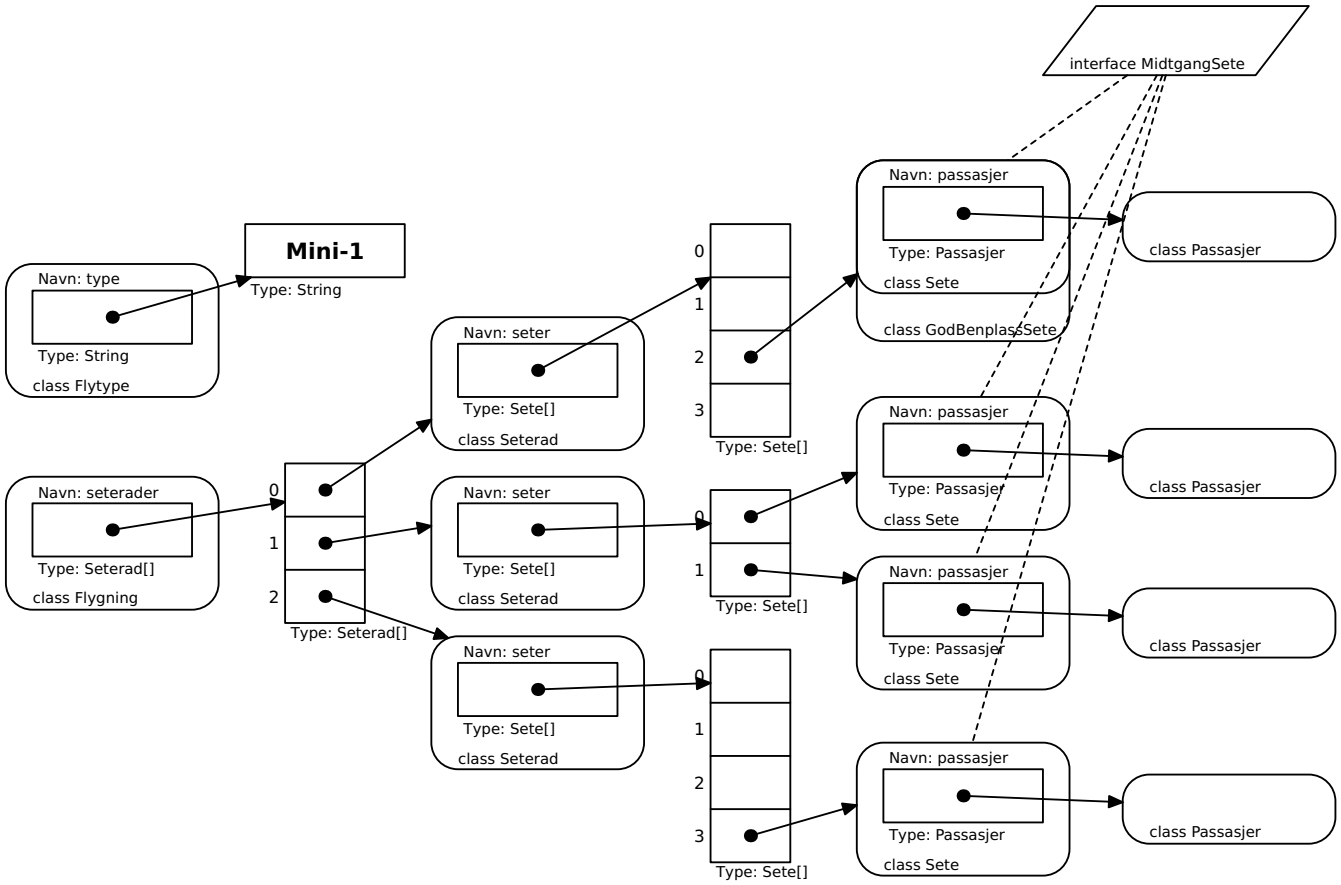
```
        } else {
            Node p  = forste;
            // Invariant: p.prioritet < ny.prioritet
            while (p.neste != null && p.neste.prioritet<ny.prioritet)
                p = p.neste;
            ny.neste = p.neste;  // NB! Fungerer også om p er sist i listen.
            p.neste = ny;
        }
    }

    public  T taUt() {
        if (forste == null) return null;

        T ut = forste.data;
        forste = forste.neste;  ant--;
        return ut;
    }

    public int antall() {
        return ant;
    }
}
```

Oppgave 3: Datastruktur



Oppgave 4: Flytype

```
class Flytype {
    String type;
    String seteinfo;

    Flytype(String id, String setekonfig) {
        type = id;  seteinfo = setekonfig;
    }

    Flygning opprettFlygning(String no) {
        return new Flygning(no, seteinfo);
```

```
    }
}
```

Oppgave 5 og 9: Flygning

```
import java.util.Iterator;

class Flygning implements Iterable<Sete> {
    String flightNo;
    Seterad[] seterader;

    Flygning(String flight, String konfig) {
        flightNo = flight;

        String[] rader = konfig.split("\\|");
        seterader = new Seterad[rader.length];
        for (int i = 0;  i < rader.length;  i++)
            seterader[i] = new Seterad(rader[i]);
    }

    boolean book(Passasjer pas) {
        for (int i = 0;  i < seterader.length;  i++)
            if (seterader[i].book(pas)) return true;

        return false;
    }

    boolean book(Passasjer pas, boolean foretrekkerVindu, boolean foretrekkerMidtgang) {
        for (int i = 0;  i < seterader.length;  i++) {
            if (seterader[i].book(pas, foretrekkerVindu, foretrekkerMidtgang))
                return true;
        }

        return false;
    }

    boolean book(Passasjer pas1, Passasjer pas2) {
        for (int i = 0;  i < seterader.length;  i++)
            if (seterader[i].book(pas1,pas2)) return true;

        return false;
    }

    public Iterator<Sete> iterator() {
        return new SeteIterator(this);
    }

    void skriv() {
        System.out.println("Flight " + flightNo);
        for (int i = 0;  i < seterader.length;  i++)
            seterader[i].skriv();
    }
}



import java.util.Iterator;

class SeteIterator implements Iterator<Sete> {
    Flygning flygning;
    int radIx = 0, seteIx = 0;

    SeteIterator(Flygning f) {
        flygning = f;
    }

    @Override
    public boolean hasNext() {
        return radIx < flygning.seterader.length;
    }

    @Override
```

```
    public Sete next() {
        Sete s = flygning.seterader[radIx].seter[seteIx];
        seteIx++;
        if (seteIx >= flygning.seterader[radIx].seter.length) {
            radIx++;  seteIx = 0;
        }
        return s;
    }
}
```

Oppgave 6: Seterad

```
class Seterad {
    int nr;
    Sete[] seter;

    Seterad(String konfig) {
        String[] data = konfig.split(":");
        nr = Integer.parseInt(data[0]);

        // Først: Hvilke seter har vi?
        String k = data[1];
        String sTab = "";
        for (int i = 0;  i < k.length();  i++) {
            char c = k.charAt(i);
            if ('A'<=c && c<='Z') sTab += c;
        }

        // Lag setene:
        seter = new Sete[sTab.length()];
        int seteIx = 0;
        for (int i = 0;  i < k.length();  i++) {
            char c = k.charAt(i);
            if ('A'<=c && c<='Z') {
                if (i<k.length()-1 && k.charAt(i+1)=='+') {
                    seter[seteIx++] = new GodBenplassSete(nr, c);  i++;
                } else {
                    seter[seteIx++] = new Sete(nr, c);
                }
            }
        }

        // Marker setene ved midtgangen:
        seteIx = 0;
        for (int i = 0;  i < k.length();  i++) {
            char c = k.charAt(i);
            if (c == '*') {
                if (seteIx > 0) seter[seteIx-1].venstreForMidtgang = true;
                if (seteIx < seter.length) seter[seteIx].hoyreForMidtgang = true;
            } else if ('A'<=c && c<='Z') {
                seteIx++;
            }
        }

        // Marker vindussetene:
        seter[0].vedVindu = seter[seter.length-1].vedVindu = true;
    }

    boolean book(Passasjer pas) {
        for (int i = 0;  i < seter.length;  i++) {
            if (seter[i].passasjer == null) {
                seter[i].passasjer = pas;
                return true;
            }
        }
        return false;
    }

    boolean book(Passasjer pas, boolean helstVindu, boolean helstMidtgang) {
        for (int i = 0;  i < seter.length;  i++) {
            Sete s = seter[i];
            if (s.passasjer == null) {
```

```
                if (helstVindu && ! s.vedVindu) continue;
                if (helstMidtgang && ! s.erVedMidtgang()) continue;
                if (pas.harLangeBen() && ! s.passerForLangeBen()) continue;
                s.passasjer = pas;
                return true;
            }
        }
        return false;
    }

    boolean book(Passasjer pas1, Passasjer pas2) {
        for (int i = 1;  i < seter.length;  i++) {
            if (seter[i-1].passasjer==null && seter[i].passasjer==null &&
                    ! seter[i].hoyreForMidtgang) {
                seter[i-1].passasjer = pas1;
                seter[i].passasjer = pas2;
                return true;
            }
        }
        return false;
    }

    void skriv() {
        for (Sete s: seter)
            s.skriv();
    }
}
```

Oppgave 7: Sete, GodBenplassSete og MidtgangSete

```
class Sete implements MidtgangSete {
    int radNr;
    char seteNr;
    Passasjer passasjer = null;
    boolean venstreForMidtgang = false, hoyreForMidtgang = false;
    boolean vedVindu = false;

    Sete(int r, char s) {
        radNr = r;  seteNr = s;
    }

    boolean passerForLangeBen() {
        return false;
    }

    public boolean erVedMidtgang() {
        return venstreForMidtgang || hoyreForMidtgang;
    }

    boolean erVindussete() {
        return vedVindu;
    }

    void skriv() {
        System.out.print("" + radNr + seteNr + "(" +
                         (erVedMidtgang() ? "M" : "") +
                         (erVindussete() ? "V" : "") +
                         "): ");
        if (passasjer != null) passasjer.skriv();
        System.out.println();
    }
}



class GodBenplassSete extends Sete {
    GodBenplassSete(int r, char s) {
        super(r, s);
    }

    boolean passerForLangeBen() {
        return true;
```

```
    }

    void skriv() {
        System.out.print("" + radNr + seteNr + "(B" +
                         (erVedMidtgang() ? "M" : "") +
                         (erVindussete() ? "V" : "") +
                         "):");
        if (passasjer != null) passasjer.skriv();
        System.out.println();
    }
}



interface MidtgangSete {
    boolean erVedMidtgang();
}
```

Oppgave 8: Passasjer

```
class Passasjer {
    String navn;
    double hoyde;

    Passasjer(String id, double h) {
        navn = id;  hoyde = h;
    }

    boolean harLangeBen() {
        return hoyde >= 190;
    }

    void skriv() {
        System.out.print(navn);
    }
}
```

Oppgave 10: Leting etter terrorist

```
import java.util.Iterator;

class Forstelinjevokter implements Runnable {
    Flygning flygning;
    String monster;
    MistenkeligePersoner mistenkte;

    Forstelinjevokter (Flygning f, MistenkeligePersoner monitor, String pat) {
        flygning = f;  mistenkte = monitor;  monster = pat;
    }

    @Override
    public void run() {
        for(Sete s: flygning) {
            Passasjer p = s.passasjer;
            if (p != null) {
                String navn = p.navn;
                int fareindeks = Stringhjelper.inneholder(navn, monster);

                System.out.println (navn + " har fareindeks " + fareindeks);
                if (fareindeks > -1) mistenkte.settInn(p,fareindeks);
            }
        }
    }
}



class Andrelinjevokter implements Runnable {
    MistenkeligePersoner fra;

    Andrelinjevokter (MistenkeligePersoner mp) {
```

```
        fra = mp;
    }

    @Override
    public void run() {
        while (true)  {
            Passasjer p = fra.taUt();
            TerrorKlasse.undersokNoye(p);
        }
    }
}


import java.util.concurrent.*;
import java.util.concurrent.locks.*;

class MistenkeligePersoner {
    Lock laas = new ReentrantLock();
    Condition ikkeTom = laas.newCondition();
    Prioritetsko<Passasjer> alle = new Prioritetsko<Passasjer>();

    void settInn (Passasjer p, int prioritet) {
        laas.lock();
        System.out.println("Setter inn " + p.navn);
        try {
            alle.settInn(p,prioritet);
            ikkeTom.signalAll();
        } finally { laas.unlock(); }
    }

    Passasjer taUt ( )  {
        laas.lock();
        try {
            while (alle.antall() == 0) {
                try {
                    ikkeTom.await();
                } catch (InterruptedException e) {
                    System.out.println("Uventet avbrudd");
                    System.exit(1);
                }
            }

            // Nå er listen ikke tom
            Passasjer pers = alle.taUt();
            System.out.println("Tar ut " + pers.navn);
            return pers;
        } finally { laas.unlock(); }
    }
}


class Terror {
    public static void main (String[] args) {
        Flytype mini1 = new Flytype("Mini-1", "2:AC*D+F+|3:C*D|4:AC*DF");
        // Annen initiering ...

        String monster = args[0];
        MistenkeligePersoner mistenkte = new MistenkeligePersoner();

        Iterator<Flygning> flyIt = alleFlygninger();
        while (flyIt.hasNext()) {
            Flygning flygning = flyIt.next();
            Runnable r = new Forstelinjevokter(flygning, mistenkte, monster);
            new Thread(r).start();
        }

        for (int ant = 0; ant < 100; ant++)  {
            Runnable s = new Andrelinjevokter(mistenkte);
            new Thread(s).start();
        }
    }
}
```