UiO : Universitetet i Oslo

KANDIDAT

15076

PRØVE

IN1010 1 Objektorientert programmering

Emnekode	IN1010
Vurderingsform	Individuell skriftlig prøve
Starttid	04.06.2018 07:00
Sluttid	04.06.2018 13:00
Sensurfrist	--
PDF opprettet	21.03.2019 09:50

Informasjon

Oppgave	Tittel	Oppgavetype
i	Om eksamen	Dokument
1	Leiting i tekst	Programmering
2	Prioritetskø	Programmering
3	Datastruktur	Muntlig
4	Flytyper	Programmering
5	Flygninger	Programmering
6	Seterader	Programmering
7	Seter	Programmering
8	Passasjerer	Programmering
9	En iterator	Programmering
10	Søk etter terrorister	Programmering

1 Leiting i tekst

Skriv en klasse **Stringhjelper** som inneholder en statisk metode kalt **inneholder**. Denne metoden skal returnere en **int** og ha to parametre, **s** og **t**, begge av type **String**. Metoden skal undersøke om teksten **t** forekommer sammenhengende i teksten **s**. Om den gjør det, skal metoden returnere indeksen (posisjonen) i **s** der **t** starter. Om **t** ikke finnes i **s**, skal metoden returnere -1.

For eksempel forekommer teksten "EI" sammenhengende i teksten "STEIN", og den starter i 2. posisjon; husk at posisjonene starter fra 0. Teksten "SEN" forekommer ikke sammenhengende i "STEIN".

I denne oppgaven får du ikke bruke andre metoder i **String**-klassen enn **charAt(i)** som henter tegnet i posisjon nr **i** i teksten, og **length()**.

```
class Stringhelper{

    public static int innholder(String s, String t){
        temp = t.charAt(0);
        int count = 0;
        for(int i = 0; i < s.length(); i++){

            if(temp == s.charAt(i)){
                count = 0;
                while(count < t.length()){
                    if(temp != s.charAt(i + count){
                        break;
                    }

                    count++;
                    temp = t.charAt(count);

                }
                if(count == t.lenght()){
                    return i;
                }
            }

        }
        return -1;

    }

}
```

Maks poeng: 5

Knytte håndtegninger til denne oppgaven?
Bruk følgende kode:

0 3 8 1 6 8 5

2 Prioritetskø

Skriv klassen **Prioritetskoe<T>** uten bruk av noe fra Javas bibliotek.

Klassen skal ha tre metoder: **settInn(T inn, int prio)** og **T taUt()** og **int antall()**. Implementasjonen skal være en lenket liste. Prioriteten er et positivt heltall eller 0. 0 er høyeste prioritet.

Den lenkede listen skal være sortert på prioritet slik at alle med høy prioritet kommer foran alle med lav prioritet.

taUt() skal helst være en veldig rask operasjon som ikke behøver lete eller gå gjennom listen, den skal bare ta ut det første elementet i listen og dermed også et av de elementene som har høyeste prioritet.

Rekkefølgen blant de elementene som har lik prioritet spiller ingen rolle.

```
class Prioritetskoe<T>{
    private Node hode;
    private Node hale;
```

```
    private int lengde = 0;

    private class Node {
        public T data;
        public int prio;
        public Node forrige = null;;
        public Node neste = null;;

        public Node(T data, int prio){
            this.data = data;
            this.prio = prio;
        }
    }

    public int antall(){
        return lengde;
    }

    //Denne metoden er for hvis man vil fjerne elementet fra koeen
    public T taUt(){
        if(lengde < 1){
            throw NoSuchElementException;
        }else{
            Node temp = hode;
            if(lengde == 1){
                hode = null;
                hale = null;
                lengde --;
                return temp.data;
            }else if(lengde == 2){
                hode = hale;
                hode.forrige = null;
                lengde --;
                return temp.data;
            }else{
                hode = temp.neste;
                hode.forrige = null;
                lengde --;
                return temp.data;
            }
        }

    }


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

        //Hvis koeen er tom
        if(lengde == 0){
            hode = ny;
            hale = ny;
            hode.neste = hale;
            hale.forrige = hode;
            lengde ++;

        //Hvis ny element er hoeyeste prio, sette helt foran med en gang
        }else if(prio == 0){
            hode.forrige = ny;
            ny.neste = hode;
            hode = ny;
            lengde ++;

        //Hvis koeen har 1 element
        }else if(lengde == 1){
            if(prio < hode.prio){
                ny.neste = hode;
                hode.forrige = ny;
                hode = ny;
            }else{
                hode.neste = ny;
                ny.forrige = hode;
                hale = ny;
            }
            lengde ++;

        //Hvis koeen har lavere prio en siste element, sette helt bakerst med engang
        }else if(prio >= hale.prio){
            hale.neste = ny;
            ny.forrige = hale;
            hale = ny;
            lengde ++;
        }

        //eller iterere til et temp-element som har lik/stoerre prio enn elementet, sette foran temp
        else{
            Node temp = hode;
            Node foran = null;
            while(prio < temp.prio){
                temp = temp.neste;
            }
            foran = temp.forrige;
            foran.neste = ny;
```

```
97              ny.forrige = foran;
98              ny.neste = temp;
99              temp.forrige = ny;
100             lengde ++;
101
102         }
103
104     }
105
106
107
108 }
```

Maks poeng: 10

Knytte håndtegninger til denne oppgaven?
Bruk følgende kode:

0 4 5 4 5 4 0

3 Datastruktur

Les vedlagte beskrivelse og finn frem til en datastruktur som er egnet til å løse oppgaven. Tegn datastrukturen med objekter og interface-er slik den ser ut for en flyvning med det lille flyet **Mini-1** som bare har tre seterader (se neste oppgave) og der fire passasjerer har reservert plass. Bruk utdelte papirark.

Maks poeng: 10

Knytte håndtegninger til denne oppgaven?
Bruk følgende kode:

1 3 4 3 9 4 7

Oppgavekode Question code	Dato Date	Emnekode Subject code	Kandidatnummer Candidate number	Oppgavenummer Question number	Sidetall Page number
1 3 4 3 9 4 7	4. juni	IN1010	15076	3	1

Tegneområde Drawing area

Jeg antar at Mini1 har de tre radene som beskrevet i oppgave 4.

Flytype: Mini1

```
Private String type: Mini1;
private String seteinfo2: A ----;
private Seterad[] radArray;
```

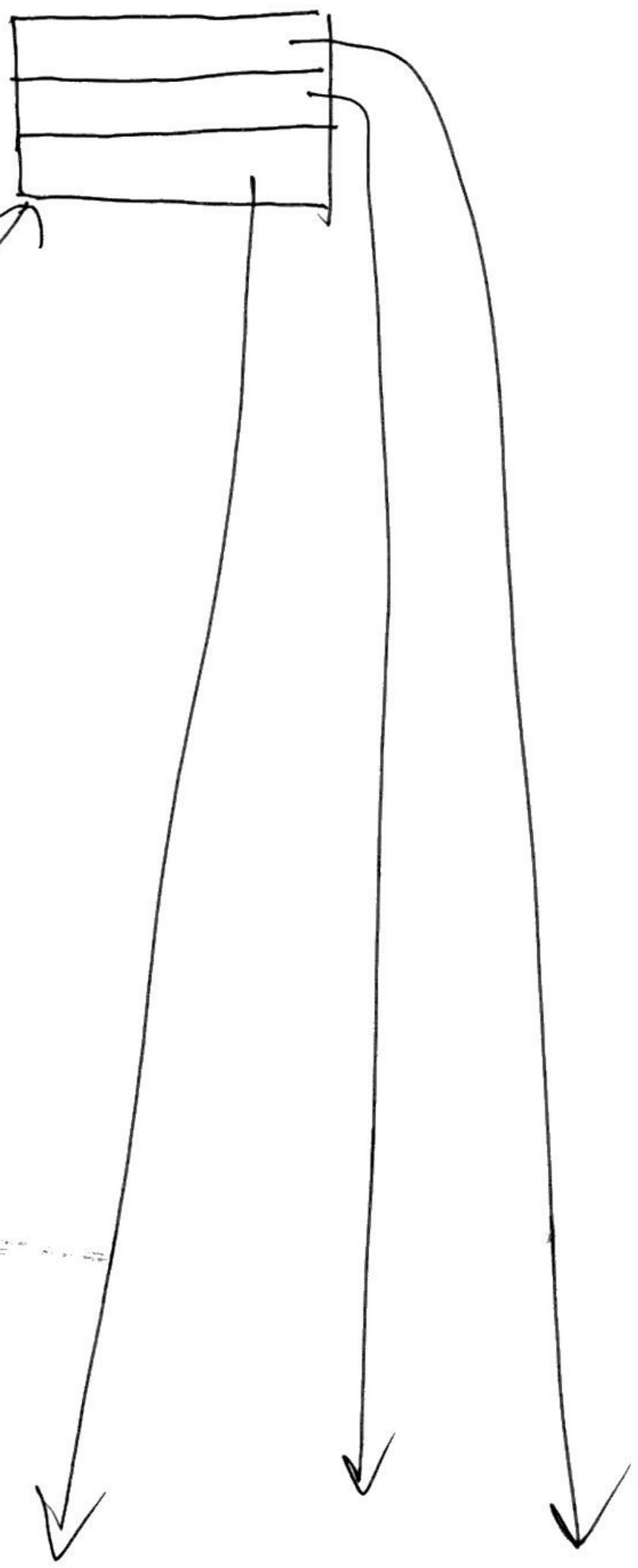
```
Public Flygning opprettflygning();
```

Flygning: Mini1 flygning

```
[illegible] static int count = [illegible];
[illegible]
private String id = 0
private String: fligheNo = Mini10
private Seterad[] radArray
```

```
Public boolean book(Passasjer Pas);
Public boolean book(Passasjer pas
                    boolean foretrekker-
                    vindu
                    boolean foretrekker-
                    Midtgang);
Public boolean book(Passajer pas,
                    Passasjer Pas2);

Public void skriv();
```



Forsetter neste side.

Oppgavekode Question code	Dato Date	Emnekode Subject code	Kandidatnummer Candidate number	Oppgavenummer Question number	Sidetall Page number
134394 7	4. juni	IN1010	15076	3	2

Tegneområde Drawing area

Seterad : 2

```
Private Sete[] seteArray;
Public int nr = 2
-----------------------
private void setNaboer();
Public Sete[] hentseteArray();
Public boolean book(Passasjer
                    pas);

Public boolean book(Passasjer
                    pas,
                    boolean foretrekk-
                      vindu,
                    boolean foretrekk-
                      midtsete);
Public boolean bookRobbel(Passasjer
                    pas1,
                    Passasjer
                    pas2);

Public String hentInfo();
```

Seterad : 3

```
private Sete[] seteArray
public int nr = 3
-----------------------
Samme
metoder
som
den
til
venstre
```

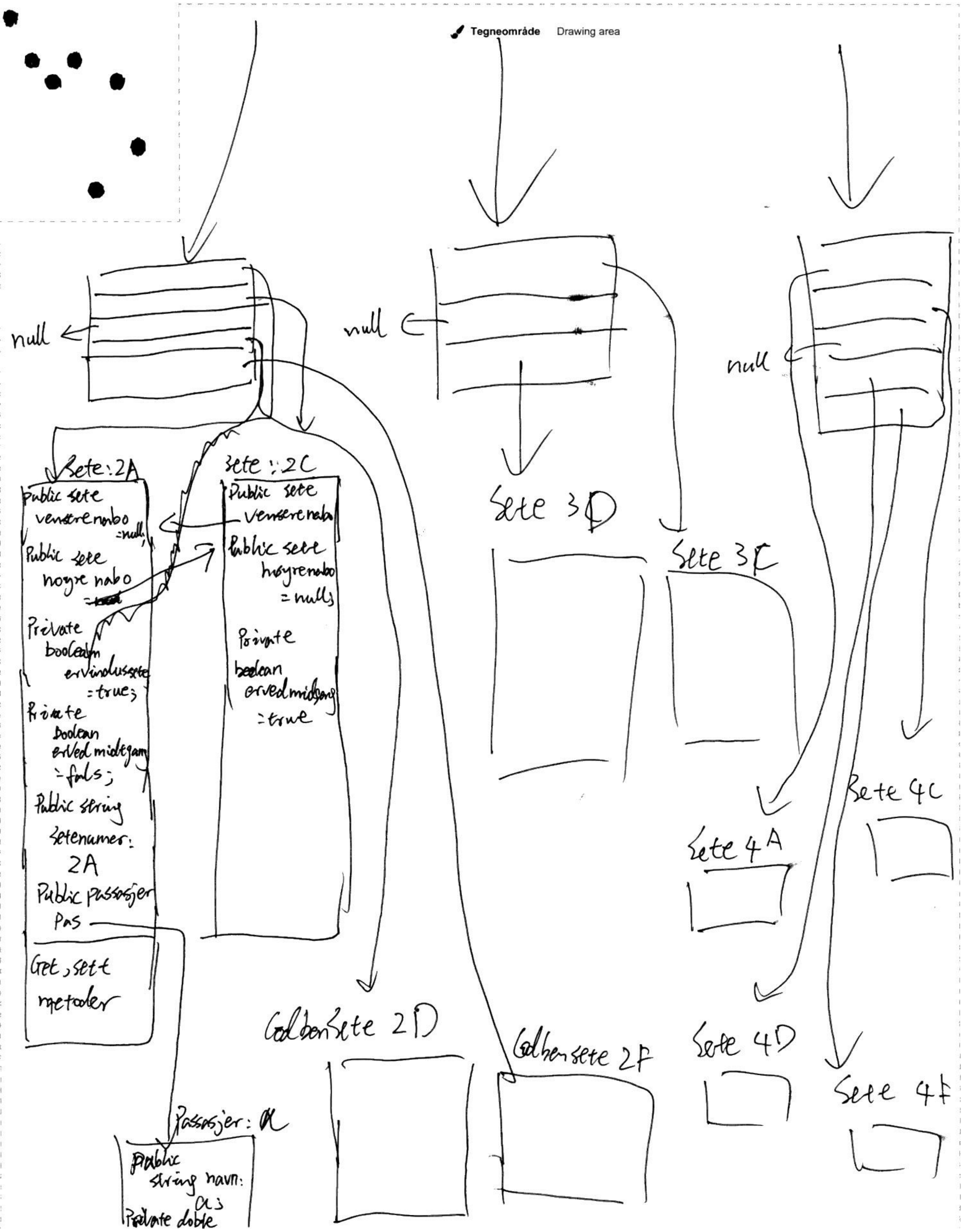
Seterad : 4

```
private Sete[] seteArray
public int nr = 4
-----------------------
Samme
metoder
som
den til
venstre
```

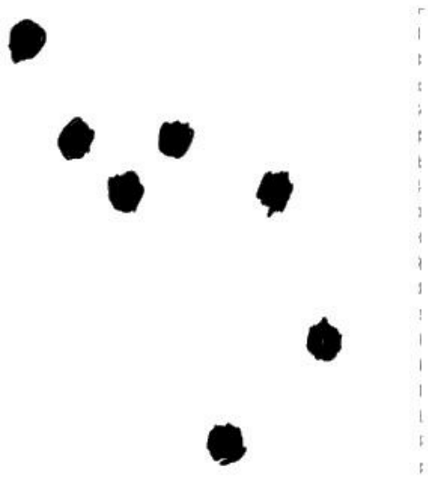
forsetter neste side

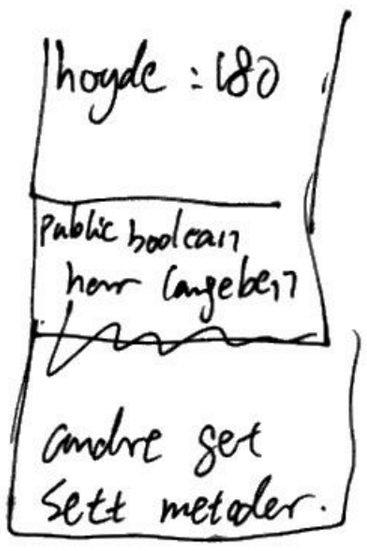
Oppgavekode Question code	Dato Date	Emnekode Subject code	Kandidatnummer Candidate number	Oppgavenummer Question number	Sidetall Page number
134347	4. juni	II010	15076	3	3

Tegneområde Drawing area



Oppgavekode	Dato	Emnekode	Kandidatnummer	Oppgavenummer	Sidetall
Question code	Date	Subject code	Candidate number	Question number	Page number
134947	4. juni	IN1010	15076	3	4





Tegneområde Drawing area

Og andre to passasjer som jeg ikke rakk å tegne.

4 Flytyper

Skriv klassen **Flytype**. Klassen skal inneholde disse elementene:

- en konstruktør
- **String type** er fabrikkens navn på flytypen.
- **String seteinfo** er en kompakt representasjon av hvilke seter som finnes i flyet: vi regner med at alle fly av samme type har samme setekonfigurasjon. For hver seterad angis radnummeret og deretter bokstavene til setene på raden. Følgende spesialtegn brukes:
  - **|** skiller seteradene
  - ***** angir en midtgang
  - **+** etter et sete angir at det har god benplass.

  Et eksempel: "2:AC*D+F+|3:C*D|4:AC*DF" angir at
  - Rad 1 finnes ikke.
  - Rad 2 har fire seter, to av dem (A og C) til venstre for midtgangen og de andre to (D og F) til høyre for midtgangen. De to sistnevnte setene har ekstra god benplass.
  - Rad 3 har kun to seter (C og D), ett på hver side av midtgangen.
  - Rad 4 har fire seter, to av dem (A og C) til venstre for midtgangen og de to andre (D og F) til høyre for midtgangen.
- **Flygning opprettFlygning()** skal lage et Flygning-objekt og initiere det med alle **Seterad** og **Sete**-objektene til (de foreløbig tomme) seter til alle passasjerene.

```java
//Jeg vil loese oppgaven slik at konfigurasjonen til flyet
//blir delt opp i hver rad. Jeg sender disse informasjon
//for hvert rad til konstruktoren til hver rad, og oppsette
//paa hvert rad blir haantert inn i rad-objektene, ikke Flytype-objektene.
//Selve Sete-objektene vil ogsaa vaere initiert i Seterad-objektene.
//Jeg gjoer dette fordi jeg mener at det er en mer objekt-orientert maate enn
//aa heller la Flytype-objektet lage oppsette til hvert rad.

class Flytype{
    private String type;
    private String seteinfo;
    private Seterad[] radArray;

    public Flytype(String type, Stirng seteinfo){
        this.type = type;
        this.seteinfo = seteinfo;
    }

    public Flygning opprettFlygning(){
        //Her blir seteInfo splittet opp i biter. Hver bit innoholder
        //Informasjon om en seterad.
        String[] radkonfig = seteinfo.split('|');
        radArray = new Seterad[radkonfig.length];

        for(int i = 0; i < radkonfig.length; i++){
            //Informasjon om rad-oppsett blir send inn i konstruktoren til Seterad-objektene
            //Sete-objektene initieres i Seterad-objektene
            radArray[i] = new Seterad(radkonfig[i]);
        }

        Flygning ny = new Flygning(this, radArray);
        return ny;
    }


}
```

Maks poeng: 12

Knytte håndtegninger til denne oppgaven?
Bruk følgende kode:

2 9 9 9 1 7 8

5 Flygninger

Skriv klassen **Flygning** som skal inneholde

- En datastruktur for alle seteradene ombord.
- En konstruktør som oppretter alle **Seterad**- og **Sete**-objektene i flyet i henhold til en **String**-spesifikasjon som angitt i forrige oppgave.
- **String flightNo** er flygningens flight-nummer.
- **boolean book(Passasjer pas)** finner et ledig sete til passasjeren og reserverer det for ham eller henne. Returverdien forteller om det var en ledig plass til vedkommende. Vi tar ikke hensyn til passasjerens høyde.
- **boolean book(Passasjer pas, boolean foretrekkerVindu, boolean foretrekkerMidtgang)** prøver å finne et egnet sete til passasjeren i henhold til vedkommendes ønsker, og at en høy passasjer får et sete med ekstra benplass. Metoden returnerer **true** hvis det var vellykket.
- **boolean book(Passasjer pas1, Passasjer pas2)** prøver å finne sete til de to passasjerene ved siden av hverandre, og ikke på hver sin side av midtgangen. Vi tar ikke hensyn til om noen av de to passasjerene er høye.
- **void skriv()** skal skrive ut (med **System.out.println**) en fullstendig passasjerliste, med navn og seteinformasjon for alle passasjerene.

```java
//Jeg antar at flightNo er satt sammen av flytype og en id av flygning-objekter
//foerste flygning til selskapet vil da vaere "FLYTYPE0"
//andre flygning til selskapet vil da vaere "FLYTYPE1" ovs.

//Flygning-objektet kaller paa Seterad-objektene sine booking relaterte metoder.


class Flygning{
    static int count = 0;
    private String id
    private Seterad[] radArray;
    private String flightNo;


    public Flygning(Flytype flytype, Seterad[] radArray){
        id = String.ValueOf(count);
        count ++;
        flightNo = flytype.type.concat(id);

        this.radArray = radArray;

    }

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

        }
        return false;
    }

    public boolean book(Passasjer pas, boolean foretrekkerVindu, boolean foretrekkerMidtgang){
            for(int i = 0; i < rad.lenth; i++){
                if(rader[i].bookVanlig(pas, foretrekkerVindu, foretrekkerMidtgang)){
                    return true;
                }
            }
            return false;
        }


//Jeg antar at to passasjerer sitter ved siden av hverandre naar de er paa samme rad. Hvis
//En pasasjer sitter i rad1 sete A og en anne passasjer sitter i rad2 sete A saa sitter de ikke
//med hverandre.
    public boolean book(Passasjer pas1, Passasjer pas2){
        for(int i = 0; i < rad.length; i ++){
            if(rader[i].bookDobbel(pas1, pas2)){
                return true;
            }
        }
        return false;
    }

    public void skriv(){
        for(int i = 0; i< rad.length; i++){
            System.out.println(rad[i].hentInfo());
        }
    }
}
```

Maks poeng: 21

Knytte håndtegninger til denne oppgaven?
Bruk følgende kode:

3 6 5 4 8 2 3

6 Seterader

Programmer klassen **Seterad** som inneholder data om én setetrad; den skal blant annet inneholde minst

- **int nr** er radens nummer.
- referanser til setene i denne raden

```java
class Seterad{
    public int nr;
    private Sete[] seterArray;

```

```
    public Seterad(String konfig){
        String temp = konfig.charAt(0);
        int antseter = konfig.lengh();


        Char check = '';

        for(int i = 2; i < konfig.length(); i++){
              check = konfig.charAt(i);

              if(check == '+'){

                 antseter --;
              }
        }

        Sete[] seteArray = new Sete[antseter];
        String thisIndex = "";
        String nextIndex = "";
        String setenummer = "";
        int infolength = konfig.length();
        for(int i = 2; i < infolength; i ++){
            if(i == infolength -1){
                thisIndex = konfig.charAt(infolength - 1);
                nextIndex = null;

                if(thisIndex == '|'){
                    seterArray[i] == null;

                }else if(thisIndex != '*'){
                    setenummer = thisIndex + String.valueOf(i);

                    if(i == 2 || i == infolength - 1){
                        if(nextIndex == '*'){
                            seteArray[i] = new GodBenplassSete(setenummer);
                            seteArray[i].setVindu();
                        }else{
                            seteArray[i] = new Sete(setenummer);
                            seteArray[i].setVindu();
                        }

                    }else{
                        if(nextIndex == '*'){
                            seteArray[i] = new GodBenplassSete(setenummer);
                        }else{
                            seteArray[i] = new Sete(setenummer);
                        }

                    }

                }else{
                    thisIndex = konfig.charAt(i);
                    nextIndex = konfig.charAt(i + 1);
                }
            }
            setNaboer();

    }

    private void setNaboer(){
        if(seteArray.length > 1){
            for(int i = 0; i < seteArray.length; i++){
                if(seteArray[i] != null){
                    if(i == 0){
                        seteArray[i].hoyreNabo = seteArray[i + 1];

                    }else if(i == seteArray.length -1){
                        seteArray[i].venstreNabo = seteArray[i-1];
                    }else{
                        seteArray[i].vestreNabo = seteArray[i -1];
                        seteArray[i].hoyreNabo = seteArray[i + 1];
                    }
                }

            }
            for(int j = 1; j < seteArray.length - 1; j++){
                if(seteArray[i].venstreNabo == null || seteArray[i].hoyreNabo == null){
                    seteArray[i].setMidtgang();
                }
            }

        }
    }

    public Sete[] hentSeteArray(){
        return seteArray;
    }

    public boolean book(Passasjer pas){
        for(int i = 0; i < seteArray.length, i++){
            if(seteArray[i] != null && seteArray[i].pas == null){
```

```
98                  seteArray[i].pas = pas;
99                  pas.setenummer = seteArray[i].setenummer;
100                 return true;
101             }
102         }
103         return false;
104 
105     }
106 
107     public boolean book(Passasjer pas, boolean foretrekkerVindu, boolean foretrekkerMidtgang){
108         boolean lang = pas.harlangeBen();
109         for(int i = 0; i < seteArray.length; i ++){
110             if(seteArray[i] != null && seteArray[i].pas == null){
111                 if(lang){
112                     if(seteArray[i] instanceof GodBenplassSete){
113                         if(foretrekkerVindu == seteArray[i].erVindussete() &&
114                             foretrekkerMidtgang == seteArray[i].erVedmidtgang()){
115 
116                             seteArray[i].pas == pas;
117                             pas.setenummer = seteArray[i].setenummer;
118                             return true;
119                         }
120                     }
121                 }
122                 else{
123                     if(foretrekkerVindu == seteArray[i].erVindussete() &&
124                         foretrekkerMidtgang == seteArray[i].erVedmidtgang()){
125                         seteArray[i].pas == pas;
126                         pas.setenummer = seteArray[i].setenummer;
127                         return true;
128 
129                     }
130                 }
131             }
132         }
133         return false;
134     }
135 
136 
137     public booean bookDobbel(passasjer pas1, Passasjer pas2){
138         for(int i = 0; i < seteArray.length; i++){
139             if(seteArray[i] != null && seteArray[i].pas == null && seteArray[i].hoeyreNabo != null){
140                 seteArray[i].pas = pas1;
141                 pas1.setenummer = seteArray[i].setenummer;
142                 seteArray[i + 1].pas = pas2;
143                 pas2.setenummer = seteArray[i+1].setenummer;
144                 return true;
145 
146             }
147         }
148         return false;
149     }
150 
151     public String hentInfo(){
152         String info = "";
153         String pasInfo = "";
154         for(int i = 0; i < array.length; i++){
155             if(seteArray[i] != null && seteArray[i].pas != null){
156                 pasInfo = pas.navn + pas.setenummer + "\n";
157                 info.concat(pasInfo);
158             }
159         }
160         return info;
161     }
162 }
```

Maks poeng: 3

Knytte håndtegninger til denne oppgaven?

Bruk følgende kode:

8 4 7 6 2 7 4

7

Seter

Skriv klassene **Sete** og **GodBenplassSete** De skal minst inneholde

- **boolean erVindussete()** angir om setet står i den ene enden av raden.
- en referanse til passasjeren som har reservert setet (eller **null** om setet ennå er ledig).

Skriv også interface-et **MidtgangSete** som skal ha

- **boolean erVedMidtgang()** angir om setet er ved midtgangen.

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


class Sete implements MidtgangSete{
    public Sete venstreNabo = null;
    public Sete hoyrenabo = null;
    private boolean erVindussete = false;
    private boolean erVedmidtgang = false;
    public String setenummer;
    public Passasjer pas = null;

    public sete(String setenummer){
        this.setenummer = setenummer;
    }

    @Override
    public boolean erVedMidtgang(){
        return erVedmidtgang;
    }

    public boolean erVindussete(){
        return erVindussete;
    }

    public void setMidtgang(){
        erVedmidtgang = true;
    }

    public void setVindu(){
        erVindussete = true;
    }

}

class GodBenplassSete extends Sete{

    public GodBenplassSete(){
        super();

    }

}
```

Maks poeng: 3

Knytte håndtegninger til denne oppgaven?

Bruk følgende kode:

7 7 1 6 8 7 9

8

Passasjerer

Skriv klassen **Passasjer**. Den skal inneholde data om passasjeren, i hvert fall

- **String navn** er passasjerens navn.
- **double hoyde** er passasjerens høyde i cm.
- **boolean harLangeBen()** forteller om passasjeren er 190 cm høy eller mer og dermed helst vil ha et sete med plass til lange ben.

```
1  class Passasjer{
2      public String navn;
3      private double hoyde;
4      public String setenummer;
5
6      public Passasjer(String navn; double hoyde){
7          this.navn = navn;
8          this.hoyde = hoyde;
9      }
10
11     public boolean harLangeBen(){
12         return hoyde >= 190;
13     }
14
15     public String hentNavn(){
16         return navn;
17     }
18
19     public double hentHoyde(){
20         return hoyde;
21     }
22 }
```

Maks poeng: 3

Knytte håndtegninger til denne oppgaven?
Bruk følgende kode:

1 0 0 5 8 9 7

9

En iterator

Anta at klassen **Flygning** endres slik at den nå **implements Iterable<Sete>** slik at vi kan iterere over alle setene i flyet på denne flygningen . Programmer klassen på nytt med denne endringen.

```
1  //Jeg antar at flightNo er satt sammen av flytype og en id av flygning-objekter
2  //foerste flygning til selskapet vil da vaere "FLYTYPE0"
3  //andre flygning til selskapet vil da vaere "FLYTYPE1" ovs.
4
5  //Flygning-objektet kaller paa Seterad-objektene sine booking relaterte metoder.
6
7  //Som foelge av implementasjon av Flygning- og Seterad-objekter, saa er min SeteIterator
8  //sannsynligvis anneledes enn det på vanlig loesningsforslaget.
9
10
11 class Flygning implements Iterable<Sete>{
12     private static int count = 0;
13     private String id
14     private Seterad[] radArray;
15     private String flightNo;
16
17     private class SeteIterator implements Iterator<Sete>{
18         int currentSeteIndex = 0;
19         int currentRadIndex = 0;
20         Sete[] seteArray = radArray[0].hentSeteArray();
21         Sete currentElement = seteArray[0];
22         Sete[] sisteArray = radArray[radArray.length - 1].hentSeteArray();
23         Sete sisteElement = sisteArray[sisteArray.length - 1];
24
25         public boolean hasNext(){
26             return currentElement != sisteElement;
27         }
28
29         public Sete next(){
```

```
            if(this.hasNext()){
                if(currentElement == seteArray[seteArray.length - 1]){
                    currentRadIndex ++;
                    seteArray = radArray[currentRadIndex];
                    Sete temp = currentElement;
                    currentSeteIndex = 0;
                    currentelement = seteArray[currentSeteIndex];
                    return temp;
                }else{
                    Sete temp = currentElement;
                    currnetSeteIndex++;
                    currentElement = seteArray[currentSeteIndex];
                    return temp;
                }
            }else{
                throw new NoSuchElementException;
            }
        }

    }


    public Flygning(String flytype, Seterad[] radArray){
        id = String.ValueOf(count);
        count ++;
        flightNo = flytype.concat(id);

        this.radArray = radArray;

    }

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

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

        }
        return false;
    }

    public boolean book(Passasjer pas, boolean foretrekkerVindu, boolean foretrekkerMidtgang){
        if(pas.harLangeBen()){
            for(int i = 0; i < rad.lenth; i++){
                if(rader[i].bookHoy(pas, foretrekkerVindu, foretrekkerMidtgang)){
                    return true;
                }
            }
            return false;
        }else{
            for(int i = 0; i < rad.lenth; i++){
                if(rader[i].bookVanlig(pas, foretrekkerVindu, foretrekkerMidtgang)){
                    return true;
                }
            }
            return false;
        }

    }
//Jeg antar at to passasjerer sitter ved siden av hverandre naar de er paa samme rad. Hvis
//En pasasjer sitter i rad1 sete A og en anne passasjer sitter i rad2 sete A saa sitter de ikke
//med hverandre.
    public boolean book(Passasjer pas1, Passasjer pas2){
        for(int i = 0; i < rad.length; i ++){
            if(rader[i].bookDobbel(pas1, pas2)){
                return true;
            }
        }
        return false;
    }

    public void skriv(){
        for(int i = 0; i< rad.length; i++){
            System.out.println(rad[i].hentInfo());
        }
    }
}
```

Maks poeng: 11

Knytte håndtegninger til denne oppgaven?
Bruk følgende kode:

0 8 3 1 7 1 2

10 Søk etter terrorister

Det er et krav at flyreservasjonsprogrammer må kunne brukes til hjelpe politiet med å finne mulige terrorister.

Du skal skrive to trådklasser samt en monitorklasse og et hovedprogram til dette bruk. Den første trådklassen skal hete **Forstelinjevokter**. Alle Forstelinjevokter-trådene skal samtidig søke gjennom hver sin flygning. Når en Forstelinjevokter finner en mistenkelig person (hva det vil si skal vi definere senere), skal tråden legge den mistenkelige personen inn i en monitor av klassen **MistenkeligePersoner**. Objekter av den andre trådklassen, kalt **Andrelinjevokter**, skal hente personer ut av denne monitoren og kalle en statisk metode i hovedprogramklassen din med navn **mistenkelig()** der disse personene kommer til å bli nøye undersøkt av politiet. (Du skal ikke skrive denne metoden.)

Hovedprogrammet skal altså brukes når politiet får tips om en sammenhengende del av navnet til en terrorist som skal ut og fly. Denne navnedelen gis som parameter til programmet og blir dermed tilgjengelig i parameter **args[0]** i hovedprogrammet når det er deklarert som

```
public static void main(String[] args) { ... }
```

La programmet gå gjennom alle flygningene og opprette og starte en Forstelinjevokter-tråd for hver flygning. For enkelthets skyld kan du anta at programmet kan få tak i en iterator over alle flygningene til flyselskapet ved å kalle den statiske metoden alleFlygninger() i hovedprogramklassen (du skal ikke programmere denne metoden).

Den sammenhengende delen av et navn som politiet får tips om, kalles et *mønster*. Jo tidligere i navnet til den mistenkte personen dette mønsteret forekommer, jo mer sannsynlig er det at personen er en terrorist. Når en Forstelinjevokter-tråd legger en mistenkt person inn i monitoren, skal avstanden fra starten av navnet til starten av mønsteret brukes som prioritet. F.eks. vil STEIN fra oppgave 1 få prioritet 2 når mønsteret er EI. Som i oppgave 2 betyr et lite tall høy prioritet.

Programmet ditt skal også starte 100 Andrelinjevokter-tråder som tar mistenkte ut av monitoren, og da skal de mistenkte med høyest prioritet tas ut først. Om monitoren er tom, skal trådene vente på at en ny person blir lagt inn.

Det er ikke nødvendig å tenke på at trådene skal terminere.

```
1  //Jeg antar at alleFlygninger() returnerer en Linkedlist lignende datasturktur, slik at jeg kan
       bruke
2  //for-each loekke.
3  //Jeg antar ogsaa at jeg kan kalle paa metodene og datastrukturene fra oppgave 1 og 2
4
5  import java.util.*;
6  import java.util.concurrent.locks.*;
7
8  class MistenkeligePersoner{
9      private ReentrantLock lock = new reentrantlock();
10     private Condition ikketomt = lock.newCondiiton();
11     Private Prioritetskoe<Passasjer> mistenkte = new Prioritetskoe<>();
12
13     public void settInn(Passasjer pas; int prio) throws InterruptedException{
14         lock.lock();
15         try{
16             mistenkte.settInn(pas, prio);
17             ikketomt.signalAll();
18
19
20         }finally{
21             lock.unlock();
22         }
23     }
24
25     public Passasjer taUt() throws InterruptedException{
26         lock.lock();
27         try{
28             if(mistenkte.antall == 0){
29                 ikketomt.await();
30             }
31             return mistenkte.taUt();
32         }finally{
33             lock.unlock();
34         }
35     }
36 }
37
38 class Forstelinjevokter implements Runneble{
39     private Flygning oppgave;
```

```
    private MistenkeligePersoner monitor;

    public Forstelinjevokter(Flygning oppgave, MistenkeligePersoner m, String check){
        this.oppgave = oppgave;
        this.monitor = m;
    }

    @Override
    public void run(){
        try{
            Passasjer p = null;
            String navn = "";
            int nivaa = -10;
            for(Sete s : oppgave){
                if(sete.pas =! null){
                    p = sete.pas;
                    navn = p.hentnavn();
                    nivaa = Stringhjelper(navn, check);
                    if(nivaa > -1){
                        monitor.settInn(p. nivaa);
                    }
                }
            }


        }catch(InterruptedException e){

        }
    }

}


class Andrelinjevokter implements Runnable{
    private MistenkeligePersoner monitor;

    public Andrelinjevokter(MistenkeligePersoner monitor){
        this.monitor = monitor;
    }

    public void run(){
        try{
            Passasjer mistenkt = null;
            while(true){
                mistenkt = monitor.taUt();
                Hovedprogram.mistenkelig(mistenkt);
            }
        }catch(InteruptedException e){

        }
    }
}

class Hovedprogram{
    static void mistenkelig(){
        //do nothing
    }

    static Linkedlist<Flygninger> alleFlygninger(){
        //returnerer alle flygningene i en linkedlist
    }

    public static void main(String[] args){
        String moenster = args[0];
        MistenkeligePersoner monitor = new MistenkeligePersoner();
        Linkedlist<Flygning> alleOppgaver = alleFlygninger();
        for(Flygning f : alleOppgave){
            new Thread ( new Forstelinjevokter(f, monitor, moenster)).start();
        }

        for(int i = 0; i < 100; i++){
            new Thread (new Andrelinjevokter(monitor)).start();
        }
    }
}
```

Maks poeng: 22

Knytte håndtegninger til denne oppgaven?
Bruk følgende kode:

9 6 1 8 0 9 3