```
public void onExecute() throws Exception {
    MyTask task = new MyTask();
    task.submit();
}

public class MyTask
implements Runnable {

    public MyTask() {
        job = new BRunnableJob(this);
    }

    public double avviksprosent; //brukes til å sette avviket, høyere
tall gir høyere tillatt verdi for hva som skal registreres som avvik se
TestVerdi() i Datapunkt.cs
    public double prosent = 0.02;

    public double proporsjonal = 1;
    public double integral = 1;
    public int pendletid = 0;

    public void submit() {
        avviksprosent = getAvviksProsent();
        job.submit(null);
    }

    int setpunktiterator;
    int datapunktiterator;

    ArrayList < Datapunkt > DataListe;
    ArrayList < Datapunkt > setpunkliste;

        public void run() {
        ArrayList < Datapunkt > bunnpunkt = new ArrayList < Datapunkt >
();
        ArrayList < Datapunkt > toppunkt = new ArrayList < Datapunkt >
();

        DataListe = new ArrayList < Datapunkt > ();
        setpunkliste = new ArrayList < Datapunkt > ();

        int tidsgrense = 120; //setter grense for hvor langt tilbake
metodene skal sjekke for pendlinger, i minutter
        long teller = 0;
        int looptellergrense = 4; // antall topper og bunner innenfor
tidsrom
        int antallPendlinger = 0; //teller for antall pendlinger (tenkt
brukt for å justere P og I verdier)
        int antallUtenfor = 0;
```

```
        double Pendlesum = 0.0; //sier noe om hvor mye pendlingene er
utenfor med, avhengig av avviksprosent


        // https://82.134.23.98:444/ord/file:%5Ependling/resultat.html

        StringBuffer outputt = new StringBuffer();
        StringBuffer debugg = new StringBuffer();
        //    {

        job.log().message("started task [" +
Thread.currentThread().getName() + "]");
        try {
            System.out.println("DEBAG 1");
            /*

DATAINNHENTING/QUERY



      */
            // Henter ut nåtid som BAbsTime.

            BAbsTime current = Clock.time();

            // Henter ut starttidspunkt som en BAbsTime, hvor vi trekker
av RelTime fra nåtid.
            BAbsTime start = current.subtract(getTimeSpan());


            // Benytter disse i en spørring.
            // Først spør vi etter verdiHistoryOrd med tid start og end.
            // Oppretter spørringen:

            BOrd qordVerdi = BOrd.make(getVerdiHistoryOrd().toString() +
"?period=timeRange;start=" + start.encodeToString() + ";end=" +
current.encodeToString() + "|bql:select * ");
            BOrd qordSettpunkt = BOrd.make(getBSPHistoryOrd().toString()
+ "?period=timeRange;start=" + start.encodeToString() + ";end=" +
current.encodeToString() + "|bql:select * ");


            // Og utfører den. Resultatet kommer ut i en table.

            BITable result = (BITable) qordVerdi.resolve().get();

            // Vi angir hvilke kolonner vi har og hva de heter.
            ColumnList columns = result.getColumns();
            Column timeColumn = columns.get(0);
```

```
            Column valueColumn = columns.get(3);
            Column statusColumn = columns.get(2);
            Column trendColumn = columns.get(1);

            // Datapunktene vi kommer til å hente ut fra tabellen vår,
verdi, status, trendstatus og tid.
            double value = 0.0;
            BStatus status = BStatus.nullStatus;
            BTrendFlags flags = null;
            BAbsTime tid = null;
            System.out.println("DEBAG 2");

            // Oppretter en stringbuffer for å lage til en output.


            // Oppretter en tableCursor som peker til hvor vi er i
tabellen.

            /*

DATAINNHENTING/LISTE




        */

            try (TableCursor c = result.cursor()) {

                // Linjen nedenfor kan tolkes som en "foreach", vi går
gjennom hele tabellen helt til det ikke er noe å gå gjennom lenger.
                while (c.next()) {

                    // For hver oppføring oppretter vi en DataPunkt.
                    value = (((BINumeric)
c.cell(valueColumn)).getNumeric());
                    tid = (BAbsTime) c.cell(timeColumn);


                    Datapunkt d = new Datapunkt(value, tid);

                    // og legger den til lista.
                    DataListe.add(d);
                }
            } catch (Exception e) {
                outputt.append("Exception for datainnhenting:" +
e.toString());
            }
```

```
            result = (BITable) qordSettpunkt.resolve().get();

            try (TableCursor c = result.cursor()) {
                columns = result.getColumns();
                timeColumn = columns.get(0);
                valueColumn = columns.get(3);
                statusColumn = columns.get(2);
                trendColumn = columns.get(1);

                while (c.next()) {


                    // For hver oppføring oppretter vi en DataPunkt.
                    value = (((BINumeric)
c.cell(valueColumn)).getNumeric());
                    tid = (BAbsTime) c.cell(timeColumn);


                    Datapunkt d = new Datapunkt(value, tid);

                    // og legger den til lista.
                    setpunkliste.add(d);
                }


                /*

OVERFØRE SETPUNKT




                */

                setpunktiterator = 0;
                datapunktiterator = 0;

                transferSp();
                /*while (datapunktiterator < DataListe.size()) // Java:
size() istedenfor Count.
          {
//            System.out.println("itererer 1");
            job.heartbeat();
            OverforeSetpunkt();
//            OverforeSetpunkt(DataListe, setpunkliste);      //se
OverføreSetpunkt()

//            OverforeSetpunkt(DataListe, setpunkliste, setpunktiterator,
```

```
datapunktiterator++);      //se OverføreSetpunkt()
// datapunktiterator++;


          }*/

            } catch (Exception e) {
                outputt.append("Exception overførsettpunkt " +
e.toString());
            }
            System.out.println("DEBAG 4");

            /*

DATAPROSESSERING


            */

            for (int j = 0; j <= DataListe.size() - 1; j++) {
                    Datapunkt d = DataListe.get(j);
                    d.TestVerdi();
                    }

            /*

            METODE 1

            */


            if (getMetode().getValue().getOrdinal() == 1) {
                for (int j = 0; j <= DataListe.size() - 1; j++) {
                    Datapunkt d = DataListe.get(j);
                    //d.TestVerdi();

                    if (j > 300) //verdi 300 er satt for at løkken ikke
skal iterere seg ut av listen
                    {
                        if (d.utenfor) // sjekker om dette og forrige
punkt er utenfor. dermed er 2 punkter etterhverandre utenfor
                        {
                            boolean ferdig = false;
                            int i = 0;
                            while (!ferdig) {
                                i++;
                                if (DataListe.get(j - i).pendling ==
true) {
                                    d.pendling = true;
```

```
                                    break;
                                }
                                if (DataListe.get(j - i).utenfor)
//sjekker om tidligere punkter også er utenfor. i metode 1 er dette
betegnet som pendling
                                {
                                    d.pendling = true;
                                    DataListe.get(j - i).pendling = true;
                                }

                                ferdig = d.SjekkTid(DataListe.get(j -
i).tid, tidsgrense);
                            }
                        }

                    }
                    teller++;
                    if (teller % 1000 == 0)
                        job.log().message("Progress: " +
String.valueOf(teller));
                }
            }


            /*

           METODE 2

            */
            else if (getMetode().getValue().getOrdinal() == 2) {
                for (int j = 0; j <= DataListe.size() - 1; j++) {
                    //DataListe.get(j).TestVerdi();
                }

                for (int j = 0; j <= DataListe.size() - 1; j++) {
                    boolean ferdig = false;
                    int i = 0;
                    int currentloop = 0;
                    if (j > DataListe.size() - 20) break;
                    if (j < 20) DataListe.get(j).utenfor = false;

                    if (DataListe.get(j).utenfor) {
                        i = 1;

                        while (!ferdig) {
                            //        iteller++;          //brukes til
testformål for å forbedre koden, teller hvor mange ganger den intærne
løkken kjører.
```

```
                            ferdig =
DataListe.get(j).SjekkTid(DataListe.get(j + i).tid, tidsgrense);
                            if (Math.abs(DataListe.get(j + i).maleverdi -
DataListe.get(j + i).setpoint) <= 0.5) {
                                currentloop++;
                                if (currentloop > looptellergrense) {
                                    DataListe.get(j).pendling = true;
//logikk for å telle topper/bunner for å spesifisere pendling?
                                    ferdig = true;
                                }
                            }

                            i++;
                            if (j > DataListe.size() - 5)
                                ferdig = true;
                        }

                        ferdig = false;
                        i = -1;

                        while (!ferdig) {
                            //         iteller++;
                            ferdig =
DataListe.get(j).SjekkTid(DataListe.get(j + i).tid, tidsgrense);
                            if (Math.abs(DataListe.get(j + i).maleverdi -
DataListe.get(j + i).setpoint) <= 0.5) {
                                currentloop++;
                                if (currentloop > looptellergrense) {
                                    DataListe.get(j).pendling = true;
                                    ferdig = true;
                                }
                            }
                            i--;
                            if (j < 5) ferdig = true;
                        }
                        teller++;
                        if (teller % 1000 == 0)
                            job.log().message(teller + " - prosess");
                    }

                }

            }
            /*

            METODE 3

                */
```

```
            else if (getMetode().getValue().getOrdinal() == 3) {

                for (int j = 0; j <= DataListe.size() - 1; j++) {

                    ArrayList < Datapunkt > tempDataListe = new ArrayList
< Datapunkt > ();

                    Datapunkt d = DataListe.get(j);

                    Datapunkt datapunktnaa = d;
                    Datapunkt datapunktforrige = d;
                    boolean ferdig = false;
                    boolean mid = false;
                    int i = 0, Toppteller = 0;
                    if (j > DataListe.size() - 5) break;
                    if (j < 10) d.utenfor = false;
                    if (d.utenfor) {
                        i = 1;
                        while (!ferdig) {
                            if (DataListe.get(j + i).pendling == true) {
                                DataListe.get(j).pendling = true;
                                ferdig = true;
                            }
                            //iteller++;          //brukes til testformål
for å forbedre koden, teller hvor mange ganger den interne løkken kjører.

                            if (mid) {
                                if (DataListe.get(j + i).utenfor) //legge
inn mer logikk for høy og lav for å karakterisere pendlingen?
                                {
                                    datapunktforrige = datapunktnaa;
                                    datapunktnaa = DataListe.get(j + i);
                                    tempDataListe.add(datapunktnaa);
                                    // Topptteller++;
                                    mid = false;
                                }
                            } else if (datapunktnaa.utenforhoy) {
                                if (DataListe.get(j + i).maleverdi <
DataListe.get(j + i).setpoint) {
                                    datapunktforrige = datapunktnaa;
                                    datapunktnaa = DataListe.get(j + i);
                                    Topptteller++;
                                    mid = true;
                                }
                            } else if (datapunktnaa.utenforlav) {
                                if (DataListe.get(j + i).maleverdi >
DataListe.get(j + i).setpoint) {
                                    datapunktforrige = datapunktnaa;
                                    datapunktnaa = DataListe.get(j + i);
```

```
                                    Topptteller++;
                                    mid = true;
                                }
                            }

                            i++;
                            if (!ferdig) ferdig =
d.SjekkTid(DataListe.get(j + i).tid, tidsgrense);

                            if (Topptteller > looptellergrense) {
                                DataListe.get(j).pendling = true;
//logikk for å telle topper/bunner for å spesifisere pendling?
                                ferdig = true;
                            }
                            if (j + i > DataListe.size() - 10) ferdig =
true;
                        }
                        ferdig = false;
                        i = -1;
                        while (!ferdig) //legg merke til at topptteller
ikke blir nullstilt.
                        {
                            if (DataListe.get(j + i).pendling == true) {
                                DataListe.get(j).pendling = true;
                                ferdig = true;
                            }
                            //                              iteller++;
    //brukes til testformål for å forbedre koden, teller hvor mange
ganger den intærne løkken kjører.
                            if (mid) {
                                if (DataListe.get(j + i).utenfor) {

                                    datapunktforrige = datapunktnaa;
                                    datapunktnaa = DataListe.get(j + i);
                                    // Topptteller++;
                                    mid = false;

                                }
                            } else if (datapunktnaa.utenforhoy) {
                                if (DataListe.get(j + i).maleverdi <
DataListe.get(j + i).setpoint) {
                                    datapunktforrige = datapunktnaa;
                                    datapunktnaa = DataListe.get(j + i);
                                    Topptteller++;
                                    mid = true;
                                }
                            } else if (datapunktnaa.utenforlav) {
                                if (DataListe.get(j + i).maleverdi >
DataListe.get(j + i).setpoint) {
```

```
                                    datapunktforrige = datapunktnaa;
                                    datapunktnaa = DataListe.get(j + i);
                                    Toppteller++;
                                    mid = true;
                                }
                            }
                            i--;
                            if (!ferdig) ferdig =
d.SjekkTid(DataListe.get(j + i).tid, tidsgrense);
                            if (Toppteller > looptellergrense) {
                                d.pendling = true; //logikk for å telle
topper/bunner for å spesifisere pendling?
                                ferdig = true;
                            }

                            if (j + i < 10) ferdig = true;
                        }


                    }
                    int dennependletid = 0;
                    if (d.pendling)
                    {
                        for (int k = 0; k <= tempDataListe.size() - 1;
k++)
                        {
                            tempDataListe.get(k).pendling = true;
 //her                            // skrive kode for å legge til
middelverdi av tid mellom punktene i listen. dette for å få frem hvor
lang tid det er mellom toppene.

                             if (k > 0)
                             {

                             BRelTime ts =
tempDataListe.get(k-1).tid.delta(tempDataListe.get(k).tid);

                             dennependletid = dennependletid +
ts.getMinutes();

                             }  //dette i minutter BabsTime.minutes?

                        }

                        if (tempDataListe.size() > 2)
                        dennependletid =
dennependletid/(tempDataListe.size()-1);

                        if (dennependletid > 0)
```

```
                        if (pendletid > 0)
                        pendletid = (pendletid + dennependletid) / 2;
//får snittet av tiden mellom toppene.

                        if (pendletid == 0) pendletid = dennependletid;

                        // outputt.append("ts: " +
String.valueOf(pendletid) + "\n");
                    }
                    teller++;
                    if (teller % 1000 == 0)
                        job.log().message(String.valueOf(teller));
                }


            }

            /*

           METODE 4

            */
            else if (getMetode().getValue().getOrdinal() == 4)

            {
                BRelTime maxtid = BRelTime.makeHours(1);
                //                TimeSpan maxtid = new TimeSpan(01, 00,
00);
                //for (int j = 0; j <= DataListe.size() - 1; j++)
                    //DataListe.get(j).TestVerdi();

                /*                foreach (Datapunkt d in DataListe)
                               {
                                   d.TestVerdi();
                               }
                */

                for (int j = 0; j <= DataListe.size() - 1; j++) {
                    Datapunkt d = DataListe.get(j);
                    int i = 0;
                    if (j > DataListe.size() - 20) break; //denne hjalp!
ellers fikk jeg feilmelding: ArgumentOutOfRangeException
                    //if (DataListe.IndexOf(d) < 20) d.Utenfor = false;
                    if (j > 300) {

                        if (d.utenforlav) {
                            boolean ferdig = false;
                            while (!ferdig) {
```

```
                                if (DataListe.get(j + i).maleverdi >
DataListe.get(j).maleverdi * (1.0 + prosent)) //sjekker om grafen stiger
med måleverdi + x% for å finne ut om d er et bunnpunkt eller bare et
punkt nedenfor setpunktet
                                {
                                    //d er et bunnpunkt
                                    bunnpunkt.add(d);
                                    ferdig = true;
                                }
                                if (DataListe.get(j + i).maleverdi <=
DataListe.get(j).maleverdi * (1.0 + prosent)) {
                                    //d er ikke et bunnpunkt og må sjekke
videre punktene i listen
                                    i++;
                                }
                                if (i >= 50 || j + i >= DataListe.size()
- 1) ferdig = true; //sjekker bare punktene som er 50 etter og stopper
før den teller forbi lengden på listen
                                //if ((DataListe[DataListe.IndexOf(d) +
i].Tid - DataListe[DataListe.IndexOf(d)].Tid) >= maxtid ||
DataListe.IndexOf(d) + i >= 49000) ferdig = true;

                            }
                        }

                        if (d.utenforhoy) {
                            boolean ferdig = false;
                            while (!ferdig) {


                                if (DataListe.get(j + i).maleverdi <
DataListe.get(j).maleverdi * (1.0 - prosent)) {
                                    //d er et toppunkt
                                    toppunkt.add(d);
                                    ferdig = true;
                                }
                                if (DataListe.get(j + i).maleverdi >=
DataListe.get(j).maleverdi * (1.0 - prosent)) {
                                    //d er ikke et toppunkt og må lete
lenger i lister.
                                    i++;
                                }
                                if (i >= 50 || j + i >= DataListe.size()
- 1) ferdig = true; //sjekker bare punktene som er 50 etter

                            }
                        }
                    }
                    teller++;
```

```
                    if (teller % 1000 == 0)
                        job.log().message(String.valueOf(teller));

                }
                outputt.append("bunnpunkter: " +
String.valueOf(bunnpunkt.size()) + "\n");

                outputt.append("topppunkter: " +
String.valueOf(toppunkt.size()) + "\n");
                //                DateTime d1 = new DateTime(2021, 02,
02, 01, 00, 00);
                //                DateTime d2 = new DateTime(2021, 02,
02, 11, 00, 00);

                //                Skrivutpunkterinnenfortiden(d1, d2,
toppunkt);
                for (int j = 0; j <= bunnpunkt.size() - 1; j++)
                    outputt.append("BP " + bunnpunkt.get(j).toString() +
"\n");
                for (int j = 0; j <= toppunkt.size() - 1; j++)
                    outputt.append("TP " + toppunkt.get(j).toString() +
"\n");


            }


            /*
SLUTT METODE 4


*/



/*

PID KODE







*/
  // programmet viser pendling --> beregne hvor mye P og I skal endres
med (evt ut fra om det er temperatur eller trykk eller lignende.)
  // --> sette en variabel i et tidsrom for å si at det er skjedd en
endring i P og I parametere. (feks hvis programmet sjekker 5 siste
dagene,
```

```
  // og endring fikser pendligen, så vil loggen fortsatt vise at det er
pendling i opp til 5 dager etter at problemet er fikset).
  // hvis det fortsatt er pendling etter endringen, så kan det settes en
ny endring. (sette endringspunkt som en datetime og kjøre koden fra det
  // tidspunktet.?)
  // *i Pendlingview få opp en promt om å utføre endring, ha en boks som
viser at det er gjort pendling.
for (int j = 0; j <= DataListe.size() - 1; j++) { // bruke dette til
beregning av ny p verdi...
                if (DataListe.get(j).pendling)
                {
                  Pendlesum += Math.abs(DataListe.get(j).maleverdi -
DataListe.get(j).setpoint);
                  antallPendlinger++;
                }
                }

//if (pendlesum > ...)


  BAbsTime tidNaa = BAbsTime.now();     //skal sette tid nå
  BAbsTime tidForSistSjekk = getInnTid();
  double forsterkning = getInnP();
  double ifaktor = getInnI();
  double dfaktor = getInnD();


  boolean sjekket = getInnSjekket(); //hente verdi for å si om
reguleringen er sjekket innen 24 timer (for å gi reguleringen nok tid til
å se om parameterene som er satt er gode nok)

  // Her kan reguleringsparametere endres.
  // hvis regulering er på temperatur på luft gjør dette med P og I (...)

  // hvis regulering er på trykk ...(...)

  //hvis vi setter en endring må denne logges (kanskje komme opp i
grafen) og eventuelt låse for ny endring innen (24 timer)



if (antallPendlinger > 40)
/*if ( sjekket == false)*/
{

    forsterkning = forsterkning;
    ifaktor = ifaktor * 0.9;
    // ifaktor = pendletid * 0.63;
    dfaktor = 0;
```

```
}


  //set tidspunkt for når systemet er sjekket.

outputt.append("PID: " + String.valueOf(forsterkning) + ", "+
String.valueOf(ifaktor) + ", "+ String.valueOf(dfaktor) + "\n");

/* if (tidForSistSjekk + 24timer > tidNaa )
 {sjekket = false}
*/
  // promt: vil du endre til disse parameterene ( P,I,D vise både
verdiene som er nå og hvilke verdier som øsnkes at skal settes)
  setUtSjekket(sjekket);
  setUtP(forsterkning);
  setUtI(ifaktor);
  setUtD(dfaktor);


/*


 OUTPUT




     */




            for (int j = 0; j <= DataListe.size() - 1; j++) {
                if (DataListe.get(j).pendling)
                {
                  Pendlesum += Math.abs(DataListe.get(j).maleverdi -
DataListe.get(j).setpoint);
                  antallPendlinger++;
                }
                if (DataListe.get(j).utenfor)
                {
                  antallUtenfor++;
                }
```

```
            }
            Pendlesum /= antallPendlinger;


            for (int j = 0; j <= DataListe.size() - 1; j++) {
                if (DataListe.get(j).pendling)
                    outputt.append(DataListe.get(j).toString() + "\n");

            }

            outputt.append("Pendletid: " + String.valueOf(pendletid) +
"\n");
            outputt.append("Antall utenfor: " +
String.valueOf(antallUtenfor) + "\n");
            outputt.append("Pendlesum: " + String.valueOf(Pendlesum) + ",
antallPendlinger: " + antallPendlinger + "\n");

            /*


   DATA FOR CHART
                    */

            StringBuffer json = new StringBuffer();
            json.append("{\"dps\":[");

            for (int i = 0; i <= DataListe.size()-1;i++)
            {
              json.append("{\n");
              json.append("\"x\":");
              // TIMESTAMP

json.append(String.valueOf(DataListe.get(i).tid.getMillis()/1000));
              /*
              json.append("\"new Date(");
              json.append(DataListe.get(i).tid.getYear());
              json.append(",");

json.append(DataListe.get(i).tid.getMonth().getMonthOfYear());
              json.append(",");
              json.append(DataListe.get(i).tid.getDay());
              json.append(",");
              json.append(DataListe.get(i).tid.getHour());
              json.append(",");
              json.append(DataListe.get(i).tid.getMinute());
              json.append(",");
              json.append(DataListe.get(i).tid.getSecond());
              json.append(")\"");
              */
```

```
                json.append("\n");

                json.append(",\n");
                json.append("\"y\":");



json.append(String.valueOf(Math.round(DataListe.get(i).maleverdi*100.0)/1
00.0));



                json.append("\n");
                json.append(",\n");

                json.append("\"sp\":");

json.append(String.valueOf(Math.round(DataListe.get(i).setpoint*100.0)/10
0.0));
                json.append(",\n");

                json.append("\"color\":");

                if (DataListe.get(i).pendling) json.append("\"#FF0000\"");
                else json.append("\"#0000FF\"");
                // PENDLING

                json.append("\n");
                json.append("}");
                if (i != DataListe.size()-1) json.append(",");


            }


            json.append("]}");


            setJsonOutput(json.toString());




        } catch (Exception ie) {
            outputt.append("XXXXX" + ie.toString() + "\n");
        }
        // Skriver til "output".
        setOutput(outputt.toString());
```

```
        setDebug(debugg.toString());

        // Vi lar jobben vår gi en heartbeat til systemet, slik at vi vet
den ikke har hengt seg opp.
        job.heartbeat();



        job.log().message("ended task [" +
Thread.currentThread().getName() + "]");
        //    }
    }


    private void transferSp() {
        int j = 0;
        for (int i = 0; i <= setpunkliste.size() - 2; i++) {
            while (setpunkliste.get(i +
1).tid.compareTo(DataListe.get(j).tid) >= 0) // Så lenge settpunktet er
foran datapunktet i tid...
            {
                if (j >= DataListe.size()-1) break;

                DataListe.get(j).setpoint =
setpunkliste.get(i).maleverdi; // Så setter vi verdi.
                DataListe.get(j++).doAvvikProsent(); // og gjør
avvikprsoent greia.
                job.heartbeat();
            }
            //    System.out.println(setpunkliste.get(i).maleverdi);
            // DataListe.get(j).setpoint = setpunkliste.get(i).maleverdi;
            job.heartbeat();

        }
        for (; j <= DataListe.size() - 1; j++) {
            DataListe.get(j).setpoint =
setpunkliste.get(setpunkliste.size() - 1).maleverdi;
            DataListe.get(j).doAvvikProsent();
            job.heartbeat();
        }
    }




    //private void OverforeSetpunkt(ArrayList<Datapunkt> dataListe,
ArrayList<Datapunkt> setpunkliste, int setpunktiterator, int
datapunktiterator)   //overfører setpunkt fra setpunktliste til
datapunktliste. logikken skal finne riktig setpunkt til riktig tid for
```

```
datapunktet
    // private void OverforeSetpunkt(ArrayList<Datapunkt> dataListe,
ArrayList<Datapunkt> setpunkliste)   //overfører setpunkt fra
setpunktliste til datapunktliste. logikken skal finne riktig setpunkt til
riktig tid for datapunktet
    //setpunkt ser etter et tidspunkt som ligger etter "denne" men før
"neste" (tegning for å forklare?)
    private void OverforeSetpunkt() {
        if (setpunktiterator >= setpunkliste.size() - 1) // Hvis
settpunktiteratoren er for høy
        {
            setpunktiterator = setpunkliste.size() - 2; // Så stilles
litt bakover.
            System.out.println("case 1");
        }

        int denne =
DataListe.get(datapunktiterator).tid.compareTo(setpunkliste.get(setpunkti
terator).tid); // Så denne er nå.
        int neste =
DataListe.get(datapunktiterator).tid.compareTo(setpunkliste.get(setpunkti
terator + 1).tid); // og neste er neste settpunkt.
        //  int denne =
setpunkliste.get(setpunktiterator).tid.compareTo(DataListe.get(datapunkti
terator).tid);
        //  int neste =
setpunkliste.get(setpunktiterator+1).tid.compareTo(DataListe.get(datapunk
titerator).tid);

        System.out.println("Denne : " + String.valueOf(denne) + ", neste:
" + String.valueOf(neste));

        // Denne == 1, neste == -1
        if (denne > 0) {
            if (neste < 0) //de fleste punktene håndteres her
            // datapunkt er seinere enn denne og tidligere neste
            {
                System.out.println("1-Datapunktiterator: " +
String.valueOf(datapunktiterator) + ", settp " +
String.valueOf(setpunktiterator));
                DataListe.get(datapunktiterator).setpoint =
setpunkliste.get(setpunktiterator).maleverdi;
                //       DataListe.get(datapunktiterator).avvikprosent =
avviksprosent;
                datapunktiterator++;
            } else if (neste > 0) // datapunkt er seinere enn denne og
neste
            {
                if (setpunktiterator + 2 >= setpunkliste.size()) // slutt
```

```
av setpunktliste, resten av datapunkt må få dette setpunktet
                {
                    System.out.println("2-Datapunktiterator: " +
String.valueOf(datapunktiterator) + ", settp " +
String.valueOf(setpunktiterator));
                    DataListe.get(datapunktiterator).setpoint =
setpunkliste.get(setpunktiterator + 1).maleverdi;
                    //
DataListe.get(datapunktiterator).avvikprosent = avviksprosent;
                    datapunktiterator++;
                }
                setpunktiterator++;
            } else if (neste == 0) //datapunkt er samme som neste
setpunkt
            {
                System.out.println("3-Datapunktiterator: " +
String.valueOf(datapunktiterator) + ", settp " +
String.valueOf(setpunktiterator));
                DataListe.get(datapunktiterator).setpoint =
setpunkliste.get(setpunktiterator + 1).maleverdi;
                DataListe.get(datapunktiterator).avvikprosent =
avviksprosent;
                setpunktiterator++;
                datapunktiterator++;
            }

        }

        if (denne < 0) {
            if (setpunktiterator + 2 >= setpunkliste.size()) {
                System.out.println("4-Datapunktiterator: " +
String.valueOf(datapunktiterator) + ", settp " +
String.valueOf(setpunktiterator));

                DataListe.get(datapunktiterator).setpoint =
setpunkliste.get(setpunktiterator + 1).maleverdi;
                DataListe.get(datapunktiterator).avvikprosent =
avviksprosent;
                datapunktiterator++;
            } else
                setpunktiterator++;
        } else if (denne == 0) {
            System.out.println("5-Datapunktiterator: " +
String.valueOf(datapunktiterator) + ", settp " +
String.valueOf(setpunktiterator));
            DataListe.get(datapunktiterator).setpoint =
setpunkliste.get(setpunktiterator).maleverdi;
            DataListe.get(datapunktiterator).avvikprosent =
avviksprosent;
```

```
            datapunktiterator++;
            setpunktiterator++;
        }
    }


    private BRunnableJob job;
}




public class Datapunkt {
    public double maleverdi;
    public BAbsTime tid;
    public double setpoint;
    public double bias;
    public boolean utenfor;
    public boolean pendling;
    public double avvikprosent;
    public double avvikverdi;
    public boolean utenforhoy;
    public boolean utenforlav;

    public Datapunkt() {

    }

    public Datapunkt(double Imaleverdi, BAbsTime Itid) {
        maleverdi = Imaleverdi;
        tid = Itid;
        avvikprosent = getAvviksProsent();
    }

    public double getAvvikprosent() {
        return avvikprosent;
    }
    public void setAvvikprosent(double value) {
        avvikprosent = value;
        avvikverdi = setpoint * avvikprosent / 100;
    }



    public void TestVerdi() {
        if (maleverdi <= setpoint - avvikverdi) {
            utenforlav = true;
            utenfor = true;
        } else if (maleverdi >= setpoint + avvikverdi) {
```

```
            utenforhoy = true;
            utenfor = true;
        } else utenfor = false;

    }

    public boolean SjekkTid(BAbsTime d, int minutter) // Hva er en
gunstig tidsforskjell å sjekke på. evt sjekke forskjellig på pendling og
uheldig regulering?
    {
        boolean InnenforTiden = false;
        BRelTime ts = tid.delta(d);
        // TimeSpan ts = this.Tid - d;

        //Console.WriteLine(ts);
        //TODO if (Math.Abs(ts.TotalMinutes) > minutter)
        if (Math.abs(ts.getMinutes()) > minutter)
            InnenforTiden = true;
        return InnenforTiden;

    }

    public void doAvvikProsent() {
        avvikverdi = setpoint * avvikprosent / 100;
    }

    public String toString() {

        String s = tid.encodeToString() + "\t verdi:" +
String.valueOf(Math.round(maleverdi*1000d)/1000d) + "\t setpunkt:" +
String.valueOf(Math.round(setpoint*1000d)/1000d) + "\t avviksverdi: " +
String.valueOf(Math.round(avvikverdi*1000d)/1000d) + ", pendling: " +
pendling;
        if (pendling) {
            s += "\t punktet viser pendling";
        }
        return s;
    }


}
```