```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Data;
using System.Threading;
using DTO;

namespace LogikLag
{
    public class Logik : ISubject, IObserver
    {
        private DatabaseAdgang Database;
        private IndhentDAQData DAQdata;
        private Nulpunktsjustering NulpunktObjekt;
        private Kalibrering KalibreringObjekt;
        private Filter FilterObj;
        private Analyse AnalyseKlasse;
        private Thread updateUI;
        private Thread updateNul;
        private Queue<double> minKø;
        private List<double> UILISTE;
        private List<IObserver> observers;
        private List<double> FiltreretListe;
        private List<double> databasetal;
        int counter;
        public double DiastoleVærdi { get; set; }
        public double SystoleVærdi { get; set; }
        public bool RadioProp { get; set; }
        private double beregnetNværdi;
        private double kalibreringKoef;
        public Logik()
        {
            Database = new DatabaseAdgang();
            DAQdata = new IndhentDAQData();
            NulpunktObjekt = new Nulpunktsjustering();
            KalibreringObjekt = new Kalibrering();
            FilterObj = new Filter();
            AnalyseKlasse = new Analyse();

            updateUI = new Thread(() => updateListe());

            UILISTE = new List<double>();
            observers = new List<IObserver>();
            FiltreretListe = new List<double>();
            databasetal = new List<double>();
            minKø = new Queue<double>(100);

            beregnetNværdi = 0.0;
            counter = 0;
            kalibreringKoef = KalibreringObjekt.Kalibrer();

            DAQdata.Attach(this);

            for (int i = 0; i < 299; i++)
            {
                UILISTE.Add(0);
            }
        }
        public void StartTraad()
        {
            updateUI.Start();
        }
        private void updateListe()
        {
            while (isRunningLogik())
            {
                if (minKø.Count > 0)
                {
                    double gennemsnitKø = minKø.Dequeue();
                    gennemsnitKø = (gennemsnitKø + beregnetNværdi) * kalibreringKoef;

                    if (counter < 300)
                    {
                        UILISTE[counter] = gennemsnitKø;
                        counter++;
                    }
                    if (counter == 299)
                    {
                        counter = 0;
                    }
                }
                if (RadioProp == false)
                {
                    Notify(FiltreringLogik(UILISTE));
                }
                else
                {
                    Notify(UILISTE);
                }
            }
            Thread.Sleep(5);
        }

        //------------------ Nulpunktsjustering ---------------
        public void StartNulPunkt()
        {
            {
                DAQdata.indhentData();
                updateNul = new Thread(() => nulpunktsJustering());
                updateNul.Start();
            }
        }
        public void nulpunktsJustering()
        {
            while (isRunningLogik())
            {
                if (minKø.Count > 0)
                {
                    DAQdata.stopReadData();
                    beregnetNværdi = NulpunktObjekt.Justering((minKø.Dequeue()));
                    minKø.Clear();
                    updateNul.Abort();
                }
                Thread.Sleep(2);
            }
        }
        //------------ ISubject -----------------
        public void Attach(IObserver observer)
        {
            observers.Add(observer);
        }

        public void Notify(List<double> data)
        {
            foreach (IObserver obs in observers)
            {
                obs.Gennemsnit(data);
            }
        }
        //------------- IObserver ------------------
        public void Gennemsnit(List<double> graf)
        {
            databasetal = graf;
            minKø.Enqueue(Convert.ToDouble(graf.Average()));
        }
        //------------ Analyse --------------------
        public void getDia()
        {
            AnalyseKlasse.Diastole(FiltreringLogik(UILISTE));
            DiastoleVærdi = AnalyseKlasse.Diastole_;
        }
        public void getSys()
        {
            AnalyseKlasse.Systole(FiltreringLogik(UILISTE));
            SystoleVærdi = AnalyseKlasse.Systole_;
        }
        //------------- Filter ----------------------
        private List<double> FiltreringLogik(List<double> data)
        {
            FiltreretListe = FilterObj.Filtrering(data);
            return FiltreretListe;
        }
        //------------- DAQ ---------------------
        public bool isRunningLogik()
        {
            return DAQdata.IsRunning();
        }

        public void indhentDataLogik()
        {
            DAQdata.indhentData();
        }

        public void stopReadDataLogik()
        {
            DAQdata.stopReadData();
            updateUI.Abort();
        }
        // ----------- Database -------------------
        public int gemData(string forsøgsnavn)
        {
            return Database.gemData(forsøgsnavn, databasetal);
        }

        public void ClearData()
        {
            databasetal.Clear();
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Blodtryksmålersystem;

namespace LogikLag
{
    class Nulpunktsjustering
    {
        private double n;
        public Nulpunktsjustering()
        {

        }
        public double Justering( double justeringværdi)
        {
          n = 0 - (justeringværdi);
            return n;
        }
    }
}

```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Configuration;

namespace LogikLag
{
    class Kalibrering
    {
        public Kalibrering()
        {
        }
        public double Kalibrer()
        {
            double kalibreringsVærdi = Convert.ToDouble(ConfigurationManager.AppSettings[         ↙
    "KalibreringsKoefficient"]);
            return kalibreringsVærdi;
        }
    }
}

```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Data;

namespace LogikLag
{
    public class Analyse
    {
        public double Diastole_ { get; set; }
        public double Systole_ { get; set; }
        public Analyse()
        {
        }
        public void Diastole(List<double> diastoleListe)
        {
            Diastole_ = diastoleListe.Min();
        }
        public void Systole(List<double> systoleListe)
        {
            Systole_ = systoleListe.Max();
        }
        public void Puls()
        {
        }
    }
}

```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LogikLag
{
    class Filter
    {
        private const int AVG_LENGTH = 5;
        public Filter()
        {
        }
        public List<double> Filtrering(List<double> data)
        {
            double sum = 0;
            List<double> avgPoints = new List<double>();
            for (int i = 0; i < data.Count()- AVG_LENGTH + 1; i++)
            {
                int innerLoopCounter = 0;
                int index = i;
                while (innerLoopCounter < AVG_LENGTH)
                {
                    sum = sum + data[index];
                    innerLoopCounter += 1;
                    index += 1;
                }
                avgPoints.Add(sum / AVG_LENGTH);
                sum = 0;
            }
            return avgPoints;
        }
    }
}

```