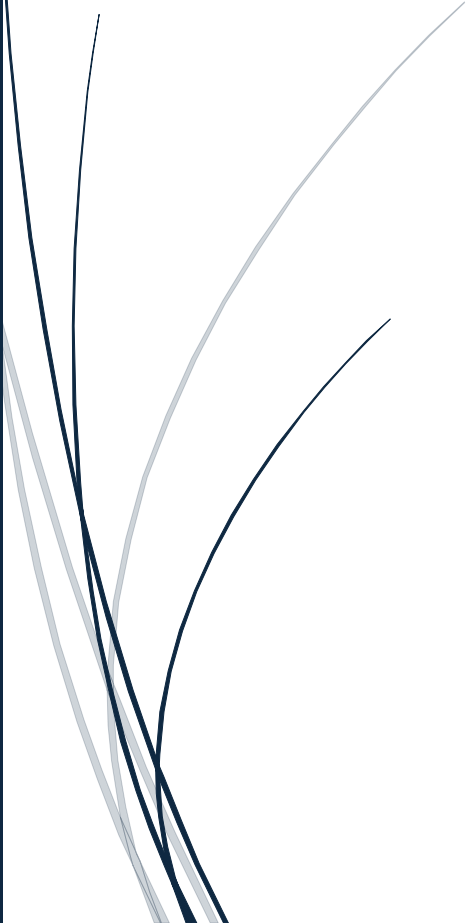


14-01-2025

Journal

ST3ITS3



Martin Büttner Rasmussen
Studienummer: 202305416

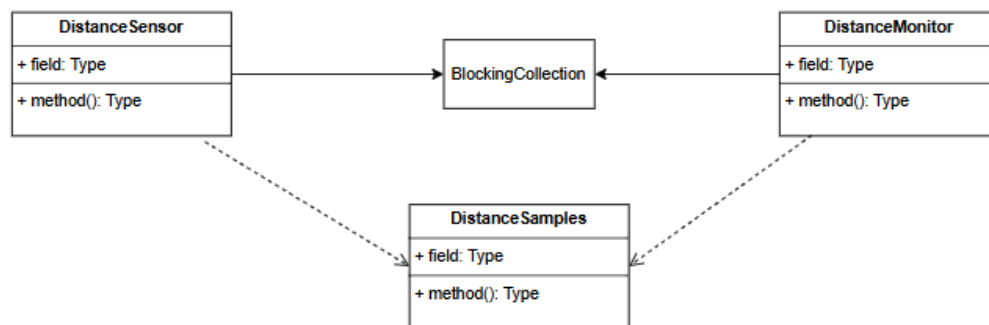
Indhold

Journal ST3ITS3:	2
Delopgave 1:	2
Delopgave 2:	4
Delopgave 3:	4
Delopgave 4:	7
Delopgave 5:	7
Delopgave 6:	8
Delopgave 7:	9
Delopgave 8:	10
Delopgave 9:	10
Delopgave 10:	11
Delopgave 11:	12
Delopgave 12	14
Delopgave 13	15
Delopgave 14	16
Delopgave 15	17
Delopgave 16	17
Delopgave 17	18
Delopgave 18	19
Delopgave 19	20
Delopgave 22	22
Delopgave 23	23
Delopgave 24	24
Delopgave 25	25
Delopgave 26	26

Journal ST3ITS3:

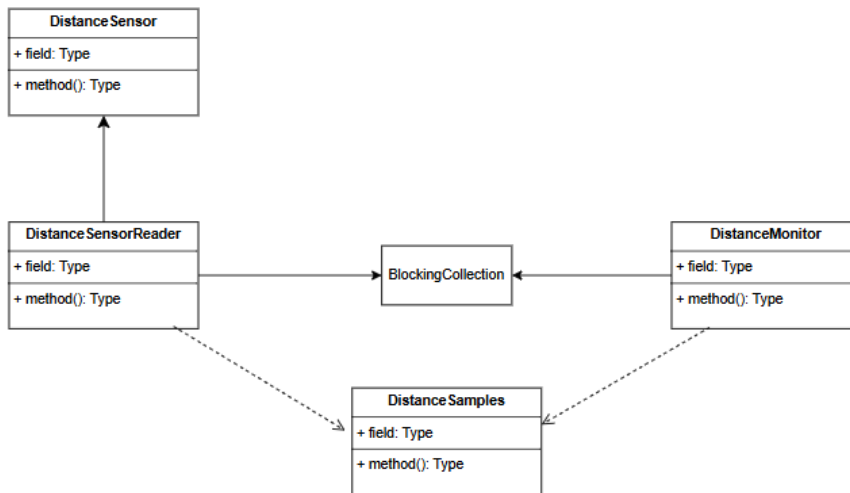
Delopgave 1:

Redegør i din journal for strukturen i et system bestående af tre klasser, **DistanceSensor**, **DistanceSample** og **DistanceMonitor**, der er forbundet med en **BlockingCollection**. Redegørelsen skal bestå af et UML klassediagram.



Figur 1 UML af systyemet opgave1

Selve strukturen af dette system bygger på producer-consumer-mønsteret, hvor vi har en producer, som er **DistanceSensor**, og en consumer, som er **DistanceMonitor**. **DistanceSensor** genererer målinger og sender dem som objekter af typen **DistanceSample** til en **BlockingCollection**. Når **DistanceSensor** er færdig med at producere en måling, sendes den til **BlockingCollection**. Herfra kan **DistanceMonitor** som consumer tilgå målingerne. Fordelen ved at bruge **BlockingCollection** er, at den er trådsikker, hvilket gør det muligt for producer og consumer at køre i hver sin tråd uden risiko for race conditions. Dette sikrer, at rækkefølgen af målingerne opretholdes, og at **DistanceSensor** ikke sender flere data, end systemet kan håndtere. Samtidig år **DistanceMonitor** at tage flere målinger, end der er produceret.

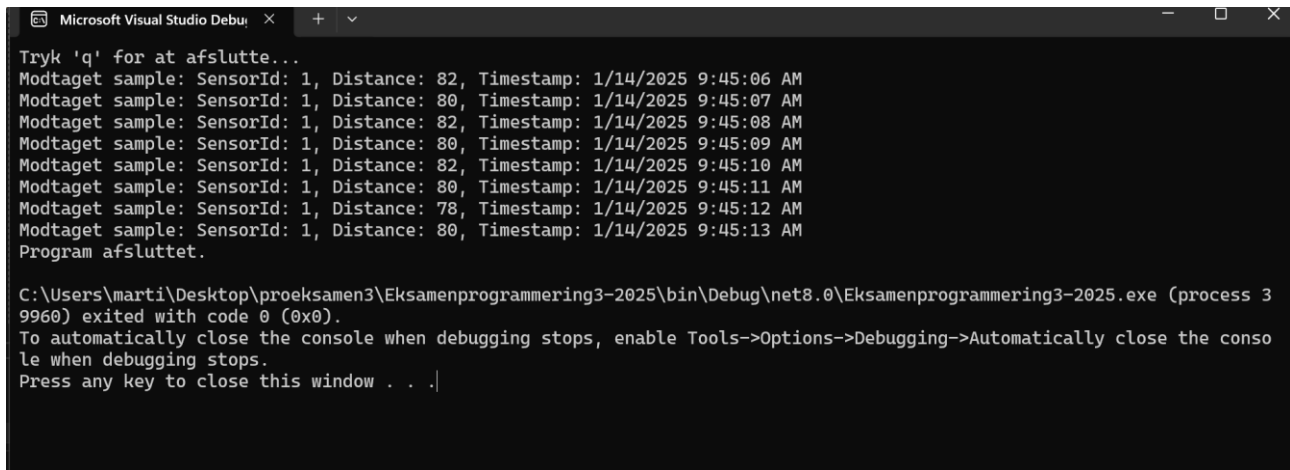


Figur 2 Udvidet system UML opgave 1

Dette system kan udvides yderligere ved at tilføje en `DistanceSensorReader`, der fungerer som en mellemmand mellem `DistanceSensor` og `BlockingCollection`. `DistanceSensorReader` læser de genererede målinger fra `DistanceSensor`, opretter et objekt af typen `DistanceSample` og sender det derefter videre til `BlockingCollection`. Denne tilføjelse reducerer koblingen mellem sensoren og resten af systemet. Hvis man på et senere tidspunkt ønsker at udskifte eller tilføje andre sensorer, kan dette implementeres uden behov for større ændringer i de eksisterende klasser.

Delopgave 2:

Implementér dit design fra Delopgave 1. Vis i din journal et eksempel på konsoloutputtet fra en kørsel af dit program, der benytter det implementerede.



```
Microsoft Visual Studio Debug Console
Tryk 'q' for at afslutte...
Modtaget sample: SensorId: 1, Distance: 82, Timestamp: 1/14/2025 9:45:06 AM
Modtaget sample: SensorId: 1, Distance: 80, Timestamp: 1/14/2025 9:45:07 AM
Modtaget sample: SensorId: 1, Distance: 82, Timestamp: 1/14/2025 9:45:08 AM
Modtaget sample: SensorId: 1, Distance: 80, Timestamp: 1/14/2025 9:45:09 AM
Modtaget sample: SensorId: 1, Distance: 82, Timestamp: 1/14/2025 9:45:10 AM
Modtaget sample: SensorId: 1, Distance: 80, Timestamp: 1/14/2025 9:45:11 AM
Modtaget sample: SensorId: 1, Distance: 78, Timestamp: 1/14/2025 9:45:12 AM
Modtaget sample: SensorId: 1, Distance: 80, Timestamp: 1/14/2025 9:45:13 AM
Program afsluttet.

C:\Users\marti\Desktop\proeksamen3\Eksamenprogrammering3-2025\bin\Debug\net8.0\Eksamenprogrammering3-2025.exe (process 39960) exited with code 0 (0x0).
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .|
```

Figur 3 Eksempel på Konsoloutputtet fra delopgave 1

Delopgave 3:

1.3 Delopgave 3

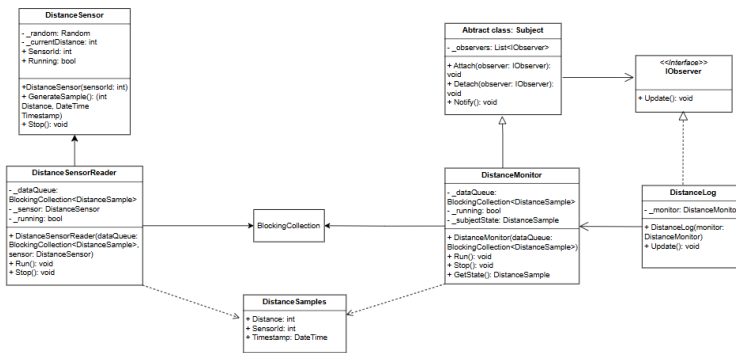
Udskriften af de modtagne `DistanceSample`'s foregår nu i `DistanceMonitor`. Denne udskrift ønskes flyttet til en `DistanceLog` klasse.

Udskriften skal indeholde alle informationer fra en `DistanceSample`. Den kunne se ud som følger

```
2025-01-03 12:34:32 – Distance: 80, Sensor: 42
2025-01-03 12:34:42 – Distance: 82, Sensor: 42
2025-01-03 12:34:52 – Distance: 84, Sensor: 42
```

Du skal anvende designmønstret GoF Observer til denne adskillelse. Redegør i din journal for denne udvidelse af dit design, i form af et UML klassediagram og UML sekvensdiagram.

Redegør i din journal ligeledes for, hvilke(t) designprincip(er) du mener overholdes bedre med denne adskillelse.



Figur 4 UML over implementering af GoF Observer mønster

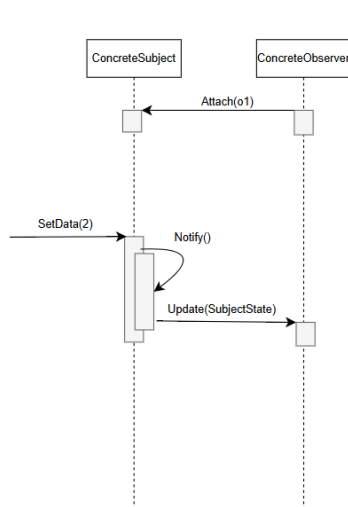
Jeg har udvidet systemet ved at implementere GoF Observer-mønsteret, hvor DistanceMonitor fungerer som et konkret subject. Det betyder, at DistanceMonitor nu har en tilstand, subjectState, og en metode GetState().

For at muliggøre observer-mønsteret er der blevet tilføjet en abstrakt klasse Subject, som indeholder en liste af observere og metoderne: Attach(Observer), Detach(Observer) og Notify().

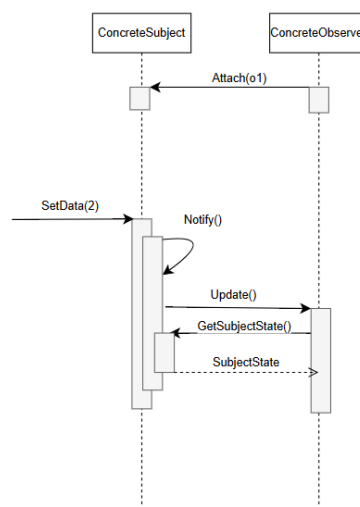
- **Attach** bruges til at tilføje observers til listen.
- **Detach** bruges til at fjerne dem.
- **Notify** kaldes, når der sker ændringer i DistanceMonitor, så alle observers bliver notificeret.

På observer-siden er der et interface IObserver, som definerer en Update()-metode. Denne metode implementeres af konkrete observers, som fx DistanceLog, der udfører forskellige handlinger, afhængigt af de data, den modtager fra DistanceMonitor.

Når systemet er sat op, og en ny måling bliver tilgængelig i DistanceMonitor, vil det ændre subjectState. Derefter kaldes Notify()-metoden, som sikrer, at alle observers på listen opdateres via deres Update()-metode



Figur 5 Sekvensdiagram for Push

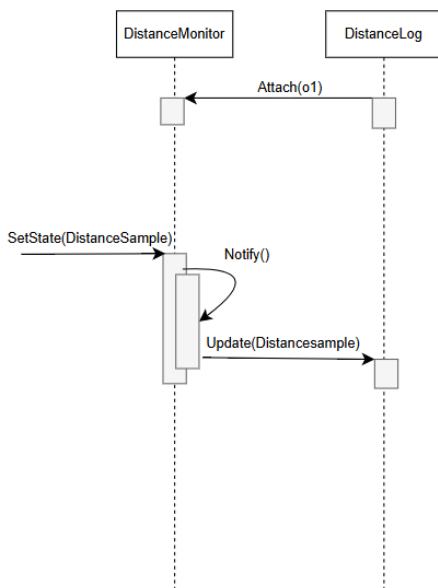


Figur 6 Sekvensdiagram for Pull

Der findes to måder at sende data mellem systemer på: **Push** og **Pull**.

I **Push-metoden** opdateres subject state, og når data er tilgængelig, kalder subject automatisk Notify(), som informerer alle observers. Observers kalder herefter deres egen Update()-metode, hvor data (f.eks. en DistanceSample) sendes direkte som en parameter. Dette gør, at observers straks modtager og håndterer data, når det er opdateret.

I **Pull-metoden** opdateres også subject state, hvorefter Notify() kaldes for at underrette alle observers. Forskellen er, at observers selv skal hente data fra subject ved hjælp af en metode som GetSubjectState(). Dette giver mere kontrol til observers, men kræver flere kald.



Figur 7 Sekvensdiagram for opgave 3

Jeg har valgt **Push-metoden**, da den gør systemet mere responsivt og sørger for, at dataen automatisk bliver sendt til observers, når den bliver opdateret. For eksempel vil DistanceMonitor opdatere sin state med SetState(DistanceSample), og derefter kalder den Notify(), som fortæller alle observers, at der er ny data. DistanceLog reagerer ved at køre sin Update()-metode, hvor den får og gemmer den nyeste måling.

Jeg mener at ved at bruge dette designmønster bliver koblingen mellem klasserne mindre, hvilket gør det lettere at tilføje nye funktioner uden at ændre i den eksisterende kode. Derudover sikrer det, at Single Responsibility-princippet overholdes, fordi hver klasse kun har ét ansvar. Det betyder, at DistanceMonitor står for overvågning og notificering, mens andre klasser håndterer deres specifikke opgaver. Dette gør systemet mere fleksibelt og skalerbart, fordi man kan tilføje nye funktioner eller klasser uden at skulle ændre på det eksisterende system.

Delopgave 4:

```
Microsoft Visual Studio Debug Console
Tryk på 'q' for at stoppe.
1/14/2025 11:09:11 AM: Distance: 82, Sensor: 1
1/14/2025 11:09:12 AM: Distance: 80, Sensor: 1
1/14/2025 11:09:13 AM: Distance: 82, Sensor: 1
1/14/2025 11:09:14 AM: Distance: 84, Sensor: 1
1/14/2025 11:09:15 AM: Distance: 86, Sensor: 1
1/14/2025 11:09:16 AM: Distance: 88, Sensor: 1
Programmet er afsluttet.

C:\Users\marti\Desktop\proeksamen3\Eksamenprogrammering3-2025\bin\Debug\net8.0\Eksamenprogrammering3-2025.exe (process 31932) exited with code 0 (0x0).
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .|
```

Figur 8 Konsoloutputet fra delopgave 3

Delopgave 5:

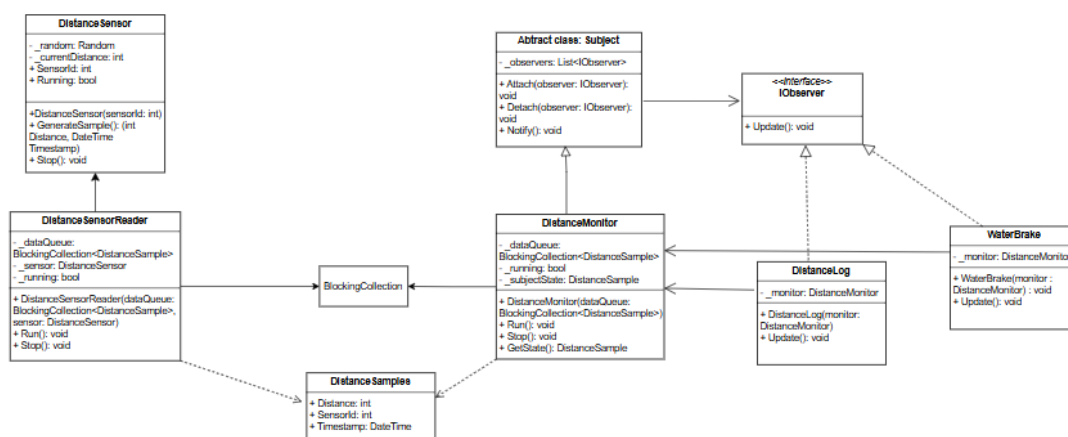
1.5 Delopgave 5

Systemet skal styre vandbremsen i afløbet.

For at simulere dette, skal du lave en `WaterBrake` klasse, som har en metode, der sætter graden af åbning mellem 0 og 100 procent. Når metoden kaldes, skal åbningsgraden skrives ud på konsollen.

Udvid dit design, så vandbremsen er helt åben, hvis afstanden til vandspejlet er mindre end 80 cm og helt lukket hvis afstanden er større end 80 cm.

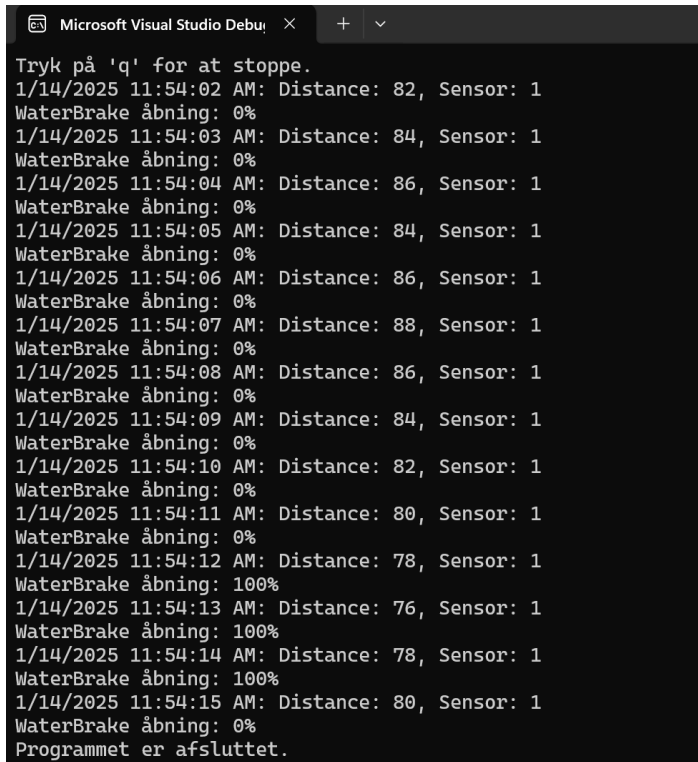
Redegør for de udvidelser til dit design som skal til, for at systemet kan overholde de nye krav. Redegørelsen skal være i form af et UML klassediagram.



Figur 9 UML til delopgave 5

Jeg har valgt at tilføje WaterBrake som en observer. WaterBrake bliver tilføjet som en observer til DistanceMonitor, så den modtager opdateringer, når der kommer nye målinger. I Update() metoden evaluerer WaterBrake, om afstanden er under 80 cm, hvor bremsen åbnes 100%, eller over 80 cm, hvor den lukkes helt til 0%. Dette printes i konsollen, så tilstanden af bremsen fremgår tydeligt. Ved at bruge Observer-mønstret adskilles ansvar, hvor DistanceMonitor kun opdaterer observers, mens WaterBrake styrer bremselogikken. Dette medfører også at man følger Single Responsibility-princippet.

Delopgave 6:



```
Microsoft Visual Studio Debu  x  +  v
Tryk på 'q' for at stoppe.
1/14/2025 11:54:02 AM: Distance: 82, Sensor: 1
WaterBrake åbning: 0%
1/14/2025 11:54:03 AM: Distance: 84, Sensor: 1
WaterBrake åbning: 0%
1/14/2025 11:54:04 AM: Distance: 86, Sensor: 1
WaterBrake åbning: 0%
1/14/2025 11:54:05 AM: Distance: 84, Sensor: 1
WaterBrake åbning: 0%
1/14/2025 11:54:06 AM: Distance: 86, Sensor: 1
WaterBrake åbning: 0%
1/14/2025 11:54:07 AM: Distance: 88, Sensor: 1
WaterBrake åbning: 0%
1/14/2025 11:54:08 AM: Distance: 86, Sensor: 1
WaterBrake åbning: 0%
1/14/2025 11:54:09 AM: Distance: 84, Sensor: 1
WaterBrake åbning: 0%
1/14/2025 11:54:10 AM: Distance: 82, Sensor: 1
WaterBrake åbning: 0%
1/14/2025 11:54:11 AM: Distance: 80, Sensor: 1
WaterBrake åbning: 0%
1/14/2025 11:54:12 AM: Distance: 78, Sensor: 1
WaterBrake åbning: 100%
1/14/2025 11:54:13 AM: Distance: 76, Sensor: 1
WaterBrake åbning: 100%
1/14/2025 11:54:14 AM: Distance: 78, Sensor: 1
WaterBrake åbning: 100%
1/14/2025 11:54:15 AM: Distance: 80, Sensor: 1
WaterBrake åbning: 0%
Programmet er afsluttet.
```

Figur 10 Konsoloutputtet fra delopgave 5

Delopgave 7:

Da mængden af vand, som løber gennem afløbet afhænger af både åbningsgraden og vandtrykket, er der brug for bedre styring af vandbremsen end blot helt lukket eller helt åben.

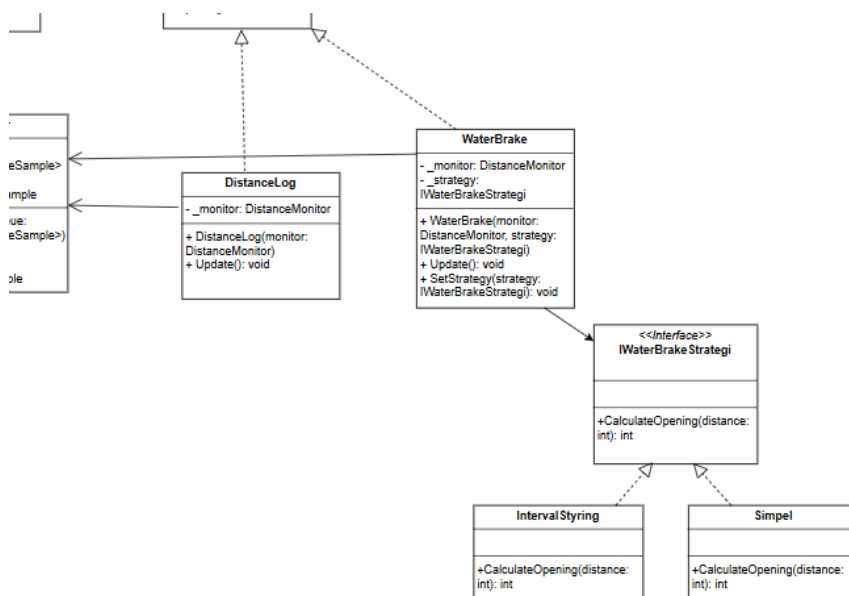
Vandbremsens åbningsgrad skal følge denne tabel:

Afstand i cm	Åbningsgrad i procent
Afstand ≥ 80	0
$80 > \text{afstand} > 61$	100
$60 > \text{afstand} > 41$	80
$40 > \text{afstand} > 21$	50
$20 > \text{afstand} \geq 0$	15

Tabel 1 - Intervalstrategi for åbningsgrad

Det skal være muligt at skifte mellem intervalstyringen og den simple åben/lukket metode til styring af vandbremsen imens programmet kører. Til dette skal du benytte GoF Strategy mønsteret.

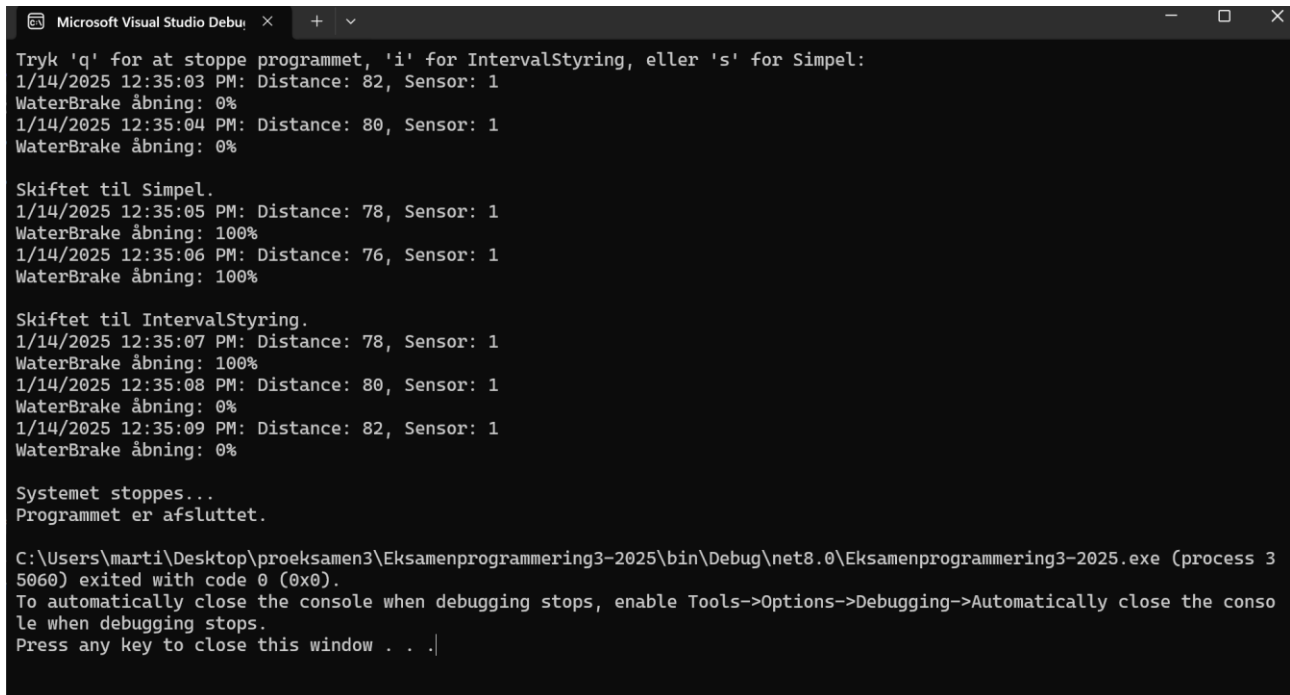
Redegør for de udvidelser til dit design som skal til, for at systemet kan overholde de nye krav. Redegørelsen skal som minimum indeholde et UML klassediagram.



Figur 11 UML til delopgave7

Jeg har brugt GoF Strategy mønsteret ved at oprette et interface, `IWaterBrakeStrategi`, som definerer metoden `CalculateOpening`. To klasser implementerer dette interface: **Simpel**, der bruger en simpel åben/lukket logik, og **IntervalStyring**, der beregner åbningen baseret på afstandsintervaller. **WaterBrake**-klassen bruger strategimønsteret til dynamisk at skifte mellem de to strategier under kørsel via `SetStrategy`. Dette medfører også at designet bliver mere fleksibelt og udvidelsesvenligt, da nye strategier kan tilføjes uden at ændre eksisterende kode.

Delopgave 8:



```
Microsoft Visual Studio Debu x + v
Tryk 'q' for at stoppe programmet, 'i' for IntervalStyring, eller 's' for Simpel:
1/14/2025 12:35:03 PM: Distance: 82, Sensor: 1
WaterBrake åbning: 0%
1/14/2025 12:35:04 PM: Distance: 80, Sensor: 1
WaterBrake åbning: 0%

Skiftet til Simpel.
1/14/2025 12:35:05 PM: Distance: 78, Sensor: 1
WaterBrake åbning: 100%
1/14/2025 12:35:06 PM: Distance: 76, Sensor: 1
WaterBrake åbning: 100%

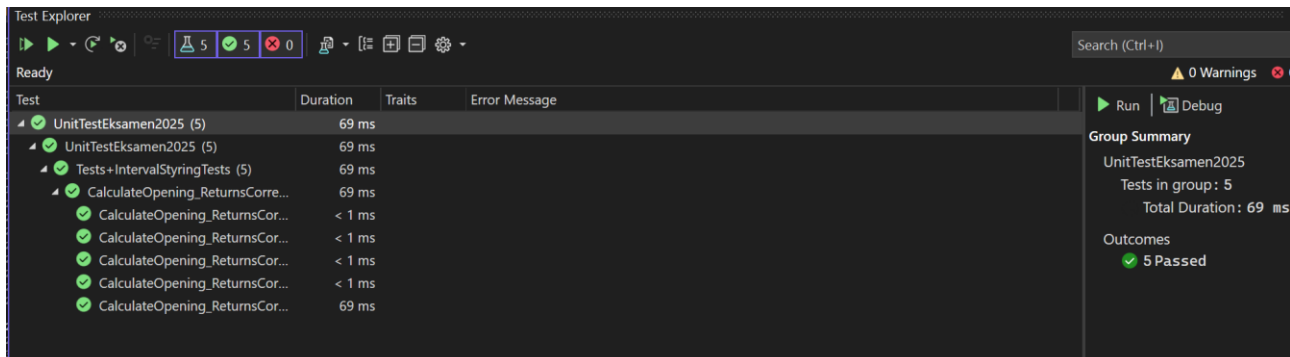
Skiftet til IntervalStyring.
1/14/2025 12:35:07 PM: Distance: 78, Sensor: 1
WaterBrake åbning: 100%
1/14/2025 12:35:08 PM: Distance: 80, Sensor: 1
WaterBrake åbning: 0%
1/14/2025 12:35:09 PM: Distance: 82, Sensor: 1
WaterBrake åbning: 0%

Systemet stoppes...
Programmet er afsluttet.

C:\Users\marti\Desktop\proeksamen3\Eksamenprogrammering3-2025\bin\Debug\net8.0\Eksamenprogrammering3-2025.exe (process 35060) exited with code 0 (0x0).
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .|
```

Figur 12 Konsoloutput for Delopgave7

Delopgave 9:



Figur 13 Screenshot af unittest

Jeg har valgt testværdierne ud fra tabel 1 i opgavebeskrivelsen, der definerer intervallerne for åbningsgraden af vandbremsen baseret på afstand i cm. For at sikre korrekt implementering af logikken, har jeg valgt én testværdi fra hver af de definerede intervaller samt en værdi uden for intervallerne (over 80 cm). Dette sikrer, at alle mulige cases bliver dækket.

De testværdier, jeg har valgt, er:

- 85 cm: For at teste, om åbningsgraden korrekt sættes til 0 %, når afstanden er over eller lig med 80 cm.
- 75 cm: For at teste intervallet $80 > \text{afstand} > 61$ cm, hvor åbningsgraden skal være 100 %.
- 50 cm: For at teste intervallet $60 > \text{afstand} > 41$ cm, hvor åbningsgraden skal være 80 %.

- 30 cm: For at teste intervallet $40 > \text{afstand} > 21$ cm, hvor åbningsgraden skal være 50 %.
- 15 cm: For at teste intervallet $20 > \text{afstand} > 0$ cm, hvor åbningsgraden skal være 15 %.

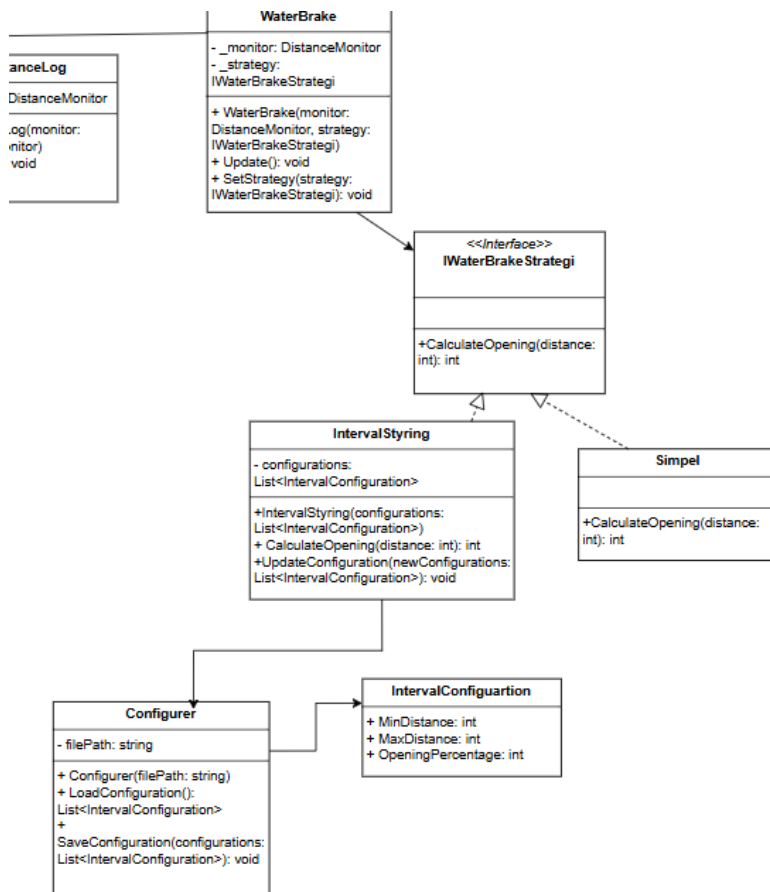
Jeg har implementeret testen som en NUnit-test ved brug af [TestCase]-attributten for at definere hver enkelt testværdi sammen med det forventede resultat. Testmetoden udfører beregningen af åbningsgraden ved at kalde metoden CalculateOpenPercentage i klassen IntervalStyring og sammenligner det beregnede resultat med det forventede resultat ved hjælp af Assert.AreEqual.

Delopgave 10:

1.10 Delopgave 10

Systemet skal kunne bruges til mange forskellige regnvandsbassiner. Derfor er der behov for, at værdierne i intervalstrategien er konfigurerbare.

Redegør i journalen for, hvordan du kan konfigurere systemet ved at benytte en JSON eller XML-fil og vis et eksempel på hvordan denne fil vil se ud. Du må gerne medtage diagrammer i redegørelsen, hvis du finder det relevant.



Figur 14 UML for delopgave 10

For at løse opgaven har jeg implementeret en **Configurer**-klasse, som håndterer læsning og skrivning af værdier fra en JSON-fil. Intervalstrategiens værdier gemmes som en liste af **IntervalConfiguration**-objekter, som indlæses dynamisk af **IntervalStyring**. Dette gør det muligt at ændre værdierne i konfigurationsfilen, så systemet kan tilpasses forskellige regnvandsbassiner uden at ændre koden.

Delopgave 11:

```
[
  {
    "MinDistance": 82,
    "MaxDistance": 2147483647,
    "OpeningPercentage": 0
  },
  {
    "MinDistance": 61,
    "MaxDistance": 81,
    "OpeningPercentage": 100
  },
  {
    "MinDistance": 41,
    "MaxDistance": 60,
    "OpeningPercentage": 80
  },
  {
    "MinDistance": 21,
    "MaxDistance": 40,
    "OpeningPercentage": 50
  },
  {
    "MinDistance": 0,
    "MaxDistance": 20,
    "OpeningPercentage": 15
  }
]
```

Figur 15 Screenshot af json-filen

```
1/14/2025 1:38:38 PM: Distance: 80, Sensor: 1
WaterBrake åbning: 100%

Skiftet til IntervalStyring.
1/14/2025 1:38:39 PM: Distance: 82, Sensor: 1
WaterBrake åbning: 0%
1/14/2025 1:38:40 PM: Distance: 84, Sensor: 1
WaterBrake åbning: 0%
1/14/2025 1:38:41 PM: Distance: 82, Sensor: 1
WaterBrake åbning: 0%
1/14/2025 1:38:42 PM: Distance: 80, Sensor: 1
WaterBrake åbning: 100%
1/14/2025 1:38:43 PM: Distance: 82, Sensor: 1
WaterBrake åbning: 0%
1/14/2025 1:38:44 PM: Distance: 84, Sensor: 1
WaterBrake åbning: 0%
1/14/2025 1:38:45 PM: Distance: 82, Sensor: 1
WaterBrake åbning: 0%
1/14/2025 1:38:46 PM: Distance: 84, Sensor: 1
WaterBrake åbning: 0%
1/14/2025 1:38:47 PM: Distance: 82, Sensor: 1
WaterBrake åbning: 0%
1/14/2025 1:38:48 PM: Distance: 84, Sensor: 1
WaterBrake åbning: 0%
1/14/2025 1:38:49 PM: Distance: 86, Sensor: 1
WaterBrake åbning: 0%
1/14/2025 1:38:50 PM: Distance: 88, Sensor: 1
WaterBrake åbning: 0%
1/14/2025 1:38:51 PM: Distance: 90, Sensor: 1
WaterBrake åbning: 0%
```

Figur 16 konsoloutput for delopgave 10

Ændrer det til at være fra 78 at den skal gå til 0%:

```
[
  {
    "MinDistance": 78,
    "MaxDistance": 2147483647,
    "OpeningPercentage": 0
  },
  {
    "MinDistance": 61,
    "MaxDistance": 77,
    "OpeningPercentage": 100
  },
  {
    "MinDistance": 41,
    "MaxDistance": 60,
    "OpeningPercentage": 80
  },
  {
    "MinDistance": 21,
    "MaxDistance": 40,
    "OpeningPercentage": 50
  },
  {
    "MinDistance": 0,
    "MaxDistance": 20,
    "OpeningPercentage": 15
  }
]
```

Figur 17 Ændret i Json-fil

Konsolvinduet:

```
WaterBrake åbning: 0%
1/14/2025 1:41:25 PM: Distance: 84, Sensor: 1
WaterBrake åbning: 0%
1/14/2025 1:41:26 PM: Distance: 86, Sensor: 1
WaterBrake åbning: 0%

Konfiguration opdateret fra fil.
1/14/2025 1:41:27 PM: Distance: 84, Sensor: 1
WaterBrake åbning: 0%
1/14/2025 1:41:28 PM: Distance: 86, Sensor: 1
WaterBrake åbning: 0%
1/14/2025 1:41:29 PM: Distance: 88, Sensor: 1
WaterBrake åbning: 0%
1/14/2025 1:41:30 PM: Distance: 86, Sensor: 1
WaterBrake åbning: 0%

Skiftet til IntervalStyring.
1/14/2025 1:41:31 PM: Distance: 84, Sensor: 1
WaterBrake åbning: 0%
1/14/2025 1:41:32 PM: Distance: 82, Sensor: 1
WaterBrake åbning: 0%
1/14/2025 1:41:33 PM: Distance: 80, Sensor: 1
WaterBrake åbning: 0%
1/14/2025 1:41:34 PM: Distance: 78, Sensor: 1
WaterBrake åbning: 0%
1/14/2025 1:41:36 PM: Distance: 80, Sensor: 1
WaterBrake åbning: 0%
1/14/2025 1:41:37 PM: Distance: 82, Sensor: 1
WaterBrake åbning: 0%
1/14/2025 1:41:38 PM: Distance: 80, Sensor: 1
WaterBrake åbning: 0%
```

Figur 18 konsoloutput efter ændring i json-fil

Delopgave 12

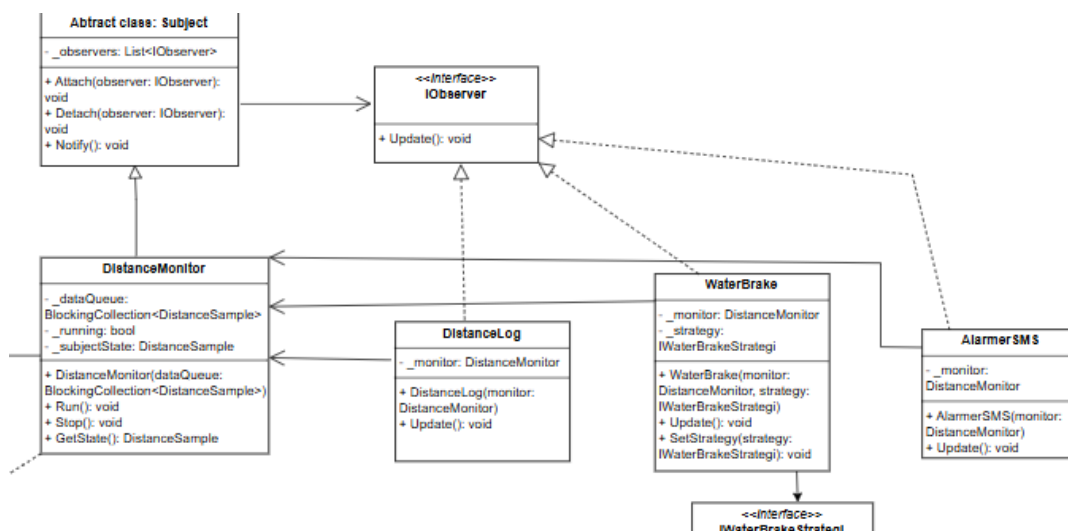
1.12 Delopgave 12

For at kunne holde øje med regnvandsbassinerne, skal systemet udvides, så det kan sende besked om vandstanden med sms.

Der er ønske om en sms til advarsel når afstanden til vandspejlet kommer ned på 25 cm og en sms med alarm når afstanden til vandspejlet kommer ned på 5 cm.

Sms-afsendelse kan simuleres med udskrift i konsollen. Udskriften skal indeholde typen (advarsel eller alarm), samt afstanden til vandspejlet.

Redegør i din journal for de udvidelser til dit design som skal til, for at systemet kan overholde de nye krav. Redegørelsen skal som minimum indeholde et UML klassesdiagram.



Figur 19 UML for Delopgave12

Jeg har tilføjet en ny observer-klasse, AlarmerSMS, der implementerer logikken for at sende advarsler og alarmer baseret på vandstanden. Klassen overvåger DistanceMonitor og udskriver enten en advarsel eller en alarm i konsollen afhængigt af vandstanden.

Delopgave 13

```
Tryk på 'q' for at stoppe programmet, 'i' for Intervals  
ation.  
1/14/2025 2:01:03 PM: Distance: 6, Sensor: 1  
WaterBrake åbning: 15%  
ADVARSEL: Vandstanden er lav (6 cm).  
1/14/2025 2:01:04 PM: Distance: 4, Sensor: 1  
WaterBrake åbning: 15%  
ALARM: Vandstanden er kritisk lav (4 cm)!  
1/14/2025 2:01:05 PM: Distance: 6, Sensor: 1  
WaterBrake åbning: 15%  
ADVARSEL: Vandstanden er lav (6 cm).  
1/14/2025 2:01:06 PM: Distance: 8, Sensor: 1  
WaterBrake åbning: 15%  
ADVARSEL: Vandstanden er lav (8 cm).  
1/14/2025 2:01:07 PM: Distance: 6, Sensor: 1  
WaterBrake åbning: 15%  
ADVARSEL: Vandstanden er lav (6 cm).  
1/14/2025 2:01:08 PM: Distance: 8, Sensor: 1  
WaterBrake åbning: 15%  
ADVARSEL: Vandstanden er lav (8 cm).  
1/14/2025 2:01:09 PM: Distance: 10, Sensor: 1  
WaterBrake åbning: 15%  
ADVARSEL: Vandstanden er lav (10 cm).  
1/14/2025 2:01:10 PM: Distance: 8, Sensor: 1  
WaterBrake åbning: 15%  
ADVARSEL: Vandstanden er lav (8 cm).
```

Figur 20 Konsoloutput for delopgave 12

Sætter Sensoren til at `_currentDistance` til at være 6, her kan vi se i konsollen at vi starter med at se en Advarsel fordi den er over 5, hvorefter den går ned på 4, hvor den udskriver ”ALARM”, som den skulle.

Delopgave 14

1.14 Delopgave 14

En afstandssensor kan være upålidelig, men man kan øge systemets samlede pålidelighed ved at benytte flere sensorer.

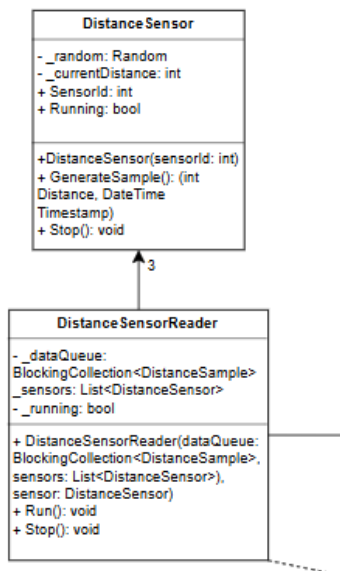
Systemet skal derfor udvides, så det indeholder 3 afstandssensorer, som kommer med hver deres måling. Middelværdien af de 2 sensorer, hvis målinger ligger tættest på hinanden skal benyttes til styring af vandbremsen.

Her er et eksempel:

Målinger: sensor-1: 50 cm, sensor-2: 60 cm, sensor-3: 54 cm

Værdi til styring = $(50 \text{ cm} + 54 \text{ cm}) / 2 = 52 \text{ cm}$

Redegør for de udvidelser til dit design som skal til, for at systemet kan overholde de nye krav. Redegørelsen skal som minimum indeholde et UML klassediagram.



Figur 21 UML for delopgave 14

Jeg har løst opgaven ved at oprette tre objekter af klassen `DistanceSensor`. Hver sensor genererer data, som sendes videre til `DistanceSensorReader`. Her sammenlignes de tre værdier, og gennemsnittet af de to målinger, der ligger tættest på hinanden, beregnes. Denne gennemsnitsværdi sættes som `DistanceSample` og sendes videre til `BlockingCollection`.

Delopgave 15

```
Tryk på 'q' for at stoppe programmet, 'i' for IntervalStyring, eller 's' for Simpel: eller 'u' for at opdatere konfiguration.
1/14/2025 2:43:39 PM: Distance: 78, Sensor: 0
WaterBrake åbning: 0%
1/14/2025 2:43:40 PM: Distance: 76, Sensor: 0
WaterBrake åbning: 100%
1/14/2025 2:43:41 PM: Distance: 76, Sensor: 0
WaterBrake åbning: 100%

Skiftet til Simpel.
1/14/2025 2:43:42 PM: Distance: 76, Sensor: 0
WaterBrake åbning: 100%
1/14/2025 2:43:43 PM: Distance: 76, Sensor: 0
WaterBrake åbning: 100%

Skiftet til IntervalStyring.
1/14/2025 2:43:44 PM: Distance: 76, Sensor: 0
WaterBrake åbning: 100%

Konfiguration opdateret fra fil.
1/14/2025 2:43:45 PM: Distance: 76, Sensor: 0
WaterBrake åbning: 100%
1/14/2025 2:43:46 PM: Distance: 78, Sensor: 0
WaterBrake åbning: 0%

Systemet stoppes...
Programmet er afsluttet.
```

Figur 22 Konsolput for delopgave 14

Der er ikke sket nogle ændringer i outputtet, da det der ændres er altså sensorværdien, sker før den bliver sat til DistanceSample.

Delopgave 16

Test	Duration	Traits	Error Message	Run Debug
UnittestDistanceSensorReader (1)	2 sec			Test Detail Summary ✓ Run_CalculatesAverageOfClosestSensors Source: UnitTest1.cs line 37 Duration: 2 sec
UnittestDistanceSensorReader (1)	2 sec			
DistanceSensorReaderTests (1)	2 sec			
Run_CalculatesAverageOfClosestSensors	2 sec			
UnitTestEksamen2025 (6)				

Figur 23 Screenshot af kørt Unittest

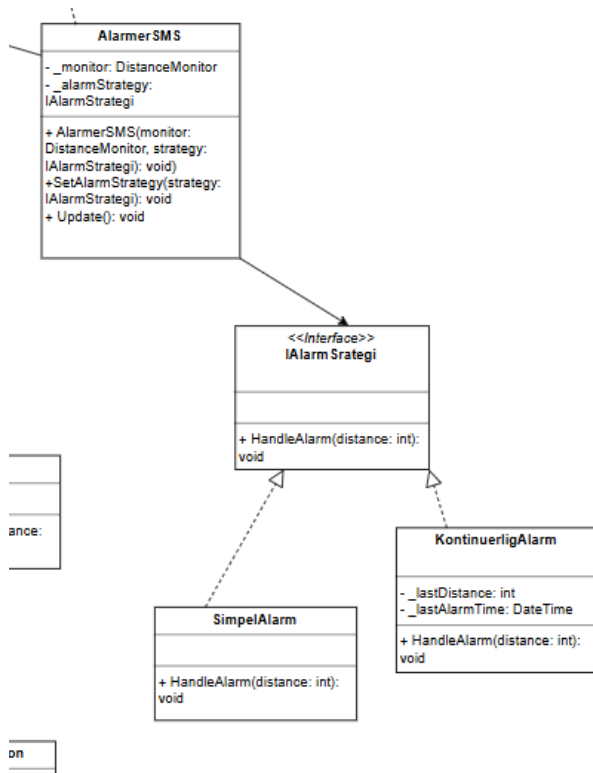
Jeg har lavet en unit test til DistanceSensorReader, hvor jeg tester beregningen af gennemsnittet af de to afstande, der er tættest på hinanden. I testen sætter jeg manuelt tre sensorer til værdierne 50, 54 og 60 og kører Run() i en tråd. Efter data er genereret, verificerer jeg, at gennemsnittet af de tætteste værdier (50 og 54) korrekt beregnes som 52 ved hjælp af Assert.AreEqual. Jeg har brugt SetUp til at initialisere sensorer og reader.

Delopgave 17

1.17 Delopgave 17

For at undgå unødvendige sms'er ønskes mulighed for at vælge en anden alarmerings-strategi, hvor den afstand til vandspejlet, som udløser advarsel eller alarm skal være til stede i 1 minut, før en alarm udsendes. Sms'en skal kun sendes igen, hvis afstanden ændrer sig.

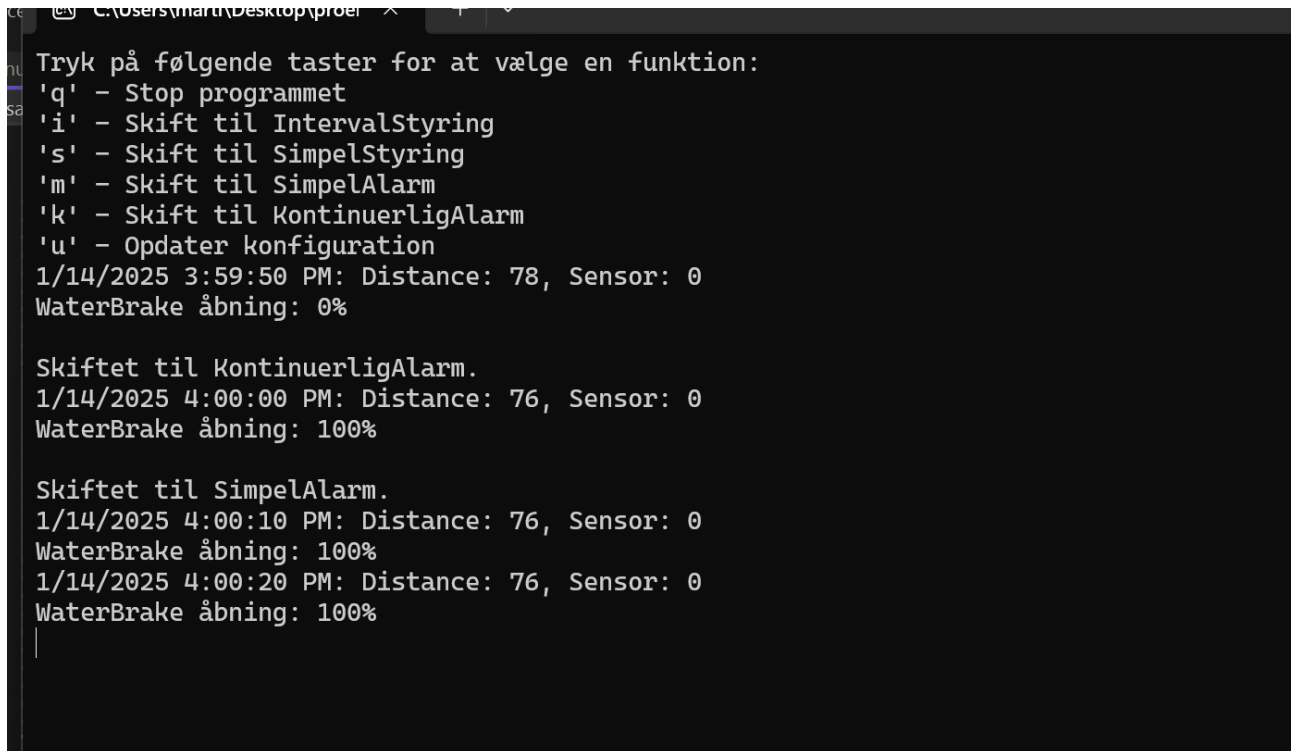
Redegør i din journal for de udvidelser som skal til, for at systemet kan overholde de nye krav.



Figur 24 UML for delopgave 17

Jeg har løst opgaven ved at bruge GoF strategi-mønsteret, hvor jeg har oprettet et interface **IAlarmStrategi**, som definerer metoden for at håndtere alarmer. Herefter har jeg implementeret to strategier: **SimpelAlarm**, som fungerer som den oprindelige logik, og **KontinuerligAlarm**, som sikrer, at alarmer kun udsendes, når afstanden ændrer sig. Klassen **AlarmerSMS** benytter dette interface og kan skifte mellem strategierne dynamisk under kørslen.

Delopgave 18

A screenshot of a terminal window with a dark background and light-colored text. The text shows a menu of options, followed by three status updates. Each update includes a timestamp, a distance value, a sensor value, and a WaterBrake status. The status changes from 0% to 100% after the first two menu selections.

```
Tryk på følgende taster for at vælge en funktion:  
'q' - Stop programmet  
'i' - Skift til IntervalStyring  
's' - Skift til SempelStyring  
'm' - Skift til SempelAlarm  
'k' - Skift til KontinuerligAlarm  
'u' - Opdater konfiguration  
1/14/2025 3:59:50 PM: Distance: 78, Sensor: 0  
WaterBrake åbning: 0%  
  
Skiftet til KontinuerligAlarm.  
1/14/2025 4:00:00 PM: Distance: 76, Sensor: 0  
WaterBrake åbning: 100%  
  
Skiftet til SempelAlarm.  
1/14/2025 4:00:10 PM: Distance: 76, Sensor: 0  
WaterBrake åbning: 100%  
1/14/2025 4:00:20 PM: Distance: 76, Sensor: 0  
WaterBrake åbning: 100%  
|
```

Figur 25 Konsoloutput for delopgave 17

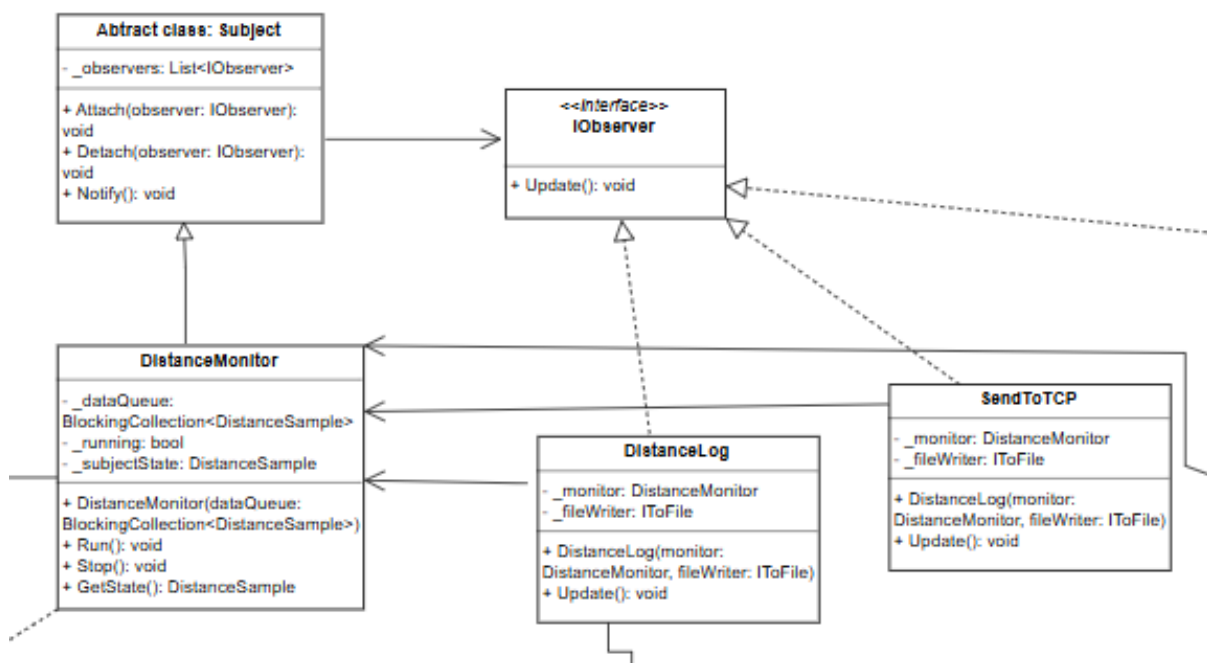
Delopgave 19

1.19 Delopgave 19

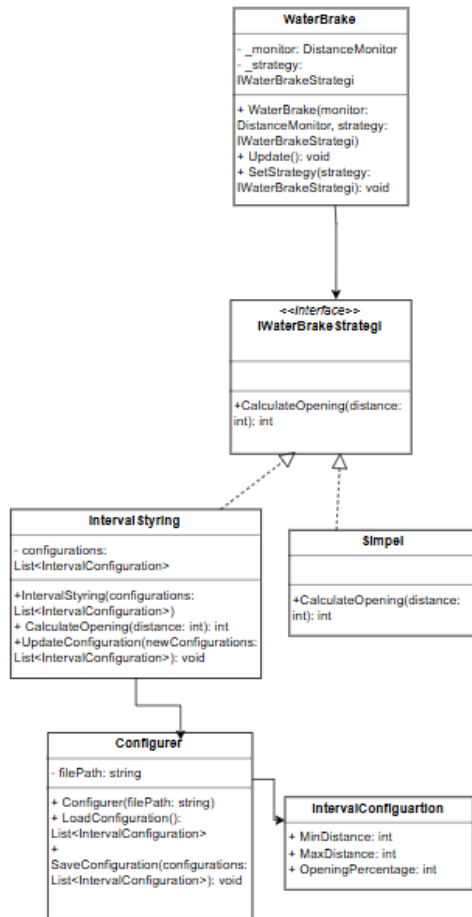
Styringen af vandbremsen skal flyttes til et separat program, hvortil der sendes beskeder om åbningsgraden af vandbremsen. Beskederne skal sendes over en TCP Socket forbindelse mellem hovedprogrammet (som du har lavet indtil nu) og programmet, som styrer vandbremsen.

Redegør for de udvidelser til dit design som skal til, for at systemet kan overholde de nye krav. Redegørelsen skal som minimum indeholde et UML klassediagram.

Det jeg ville gøre var at lave en ny observer, som skulle have til formål at modtage DistanceSample fra subjected, herefter vil den skulle sende disse målinger over en TCP Socket forbindelse fra mit hovedprogram over til et andet program (anden solution), hvor jeg ville have min WaterBrake-Del, som så ville modtage dette data fra observeren, hvor målingen ville køre igennem WaterBrake-Delens logik, hvorefter der vil blive sendt et svar tilbage gennem Tcp forbindelsen, hvor det ville blive udskrevet i konsollen



Figur 26 UML for delopgave 19



Figur 27 UML for WaterBrake-Del

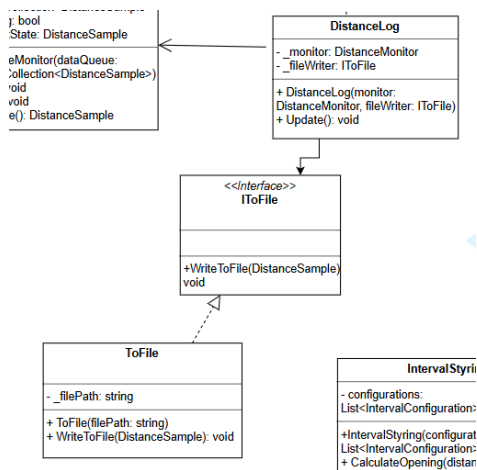
Delopgave 22

1.22 Delopgave 22

Det ønskes at de målte værdier i `DistanceSample`'s ud over at blive printet også skal gemmes i en fil-log.

Filen skal være en almindelig tekstfil (ende på .txt) og indeholde en linje for hver `DistanceSample` der er modtaget.

Udvid dit UML klassediagram med de klasser du finder nødvendigt for denne udvidelse og vis klassediagrammet i journalen.



Figur 28 UML for delopgave 22

Her har jeg igen brugt GoF Strategi-mønsteret. Jeg har lavet et interface `IToFile`, som definerer metoden `WriteToFile`, der bruges til at skrive data til en fil. Herefter har jeg implementeret `IToFile` i klassen `ToFile`, hvor selve logikken for at skrive `DistanceSample` til en tekstfil ligger. `DistanceLog` bruger dette interface til at skrive data, hvilket gør det nemt at ændre eller udvide måden, data bliver logget på, uden at det påvirker resten af systemet.

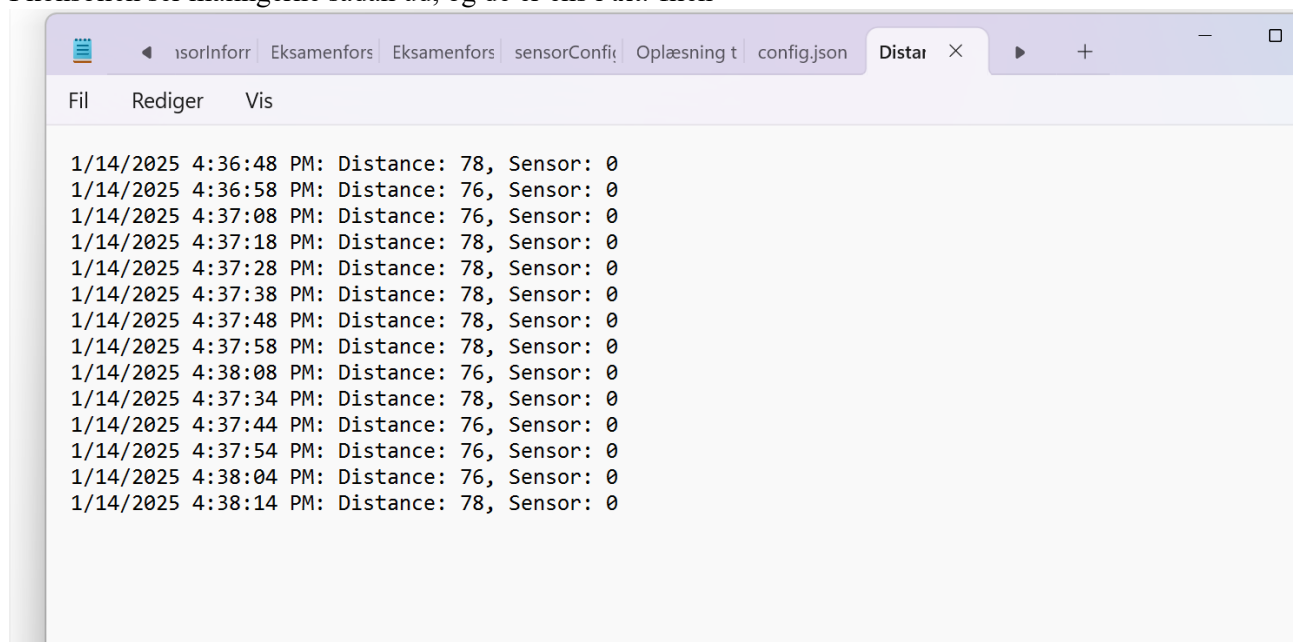
Delopgave 23

```
Tryk på følgende taster for at vælge en funktion:
'q' - Stop programmet
'i' - Skift til IntervalStyring
's' - Skift til SimpelStyring
'm' - Skift til SimpelAlarm
'k' - Skift til KontinuerligAlarm
'u' - Opdater konfiguration
1/14/2025 4:37:34 PM: Distance: 78, Sensor: 0
WaterBrake åbning: 0%
1/14/2025 4:37:44 PM: Distance: 76, Sensor: 0
WaterBrake åbning: 100%
1/14/2025 4:37:54 PM: Distance: 76, Sensor: 0
WaterBrake åbning: 100%
1/14/2025 4:38:04 PM: Distance: 76, Sensor: 0
WaterBrake åbning: 100%
1/14/2025 4:38:14 PM: Distance: 78, Sensor: 0
WaterBrake åbning: 0%

Systemet stoppes...
Programmet er afsluttet.
```

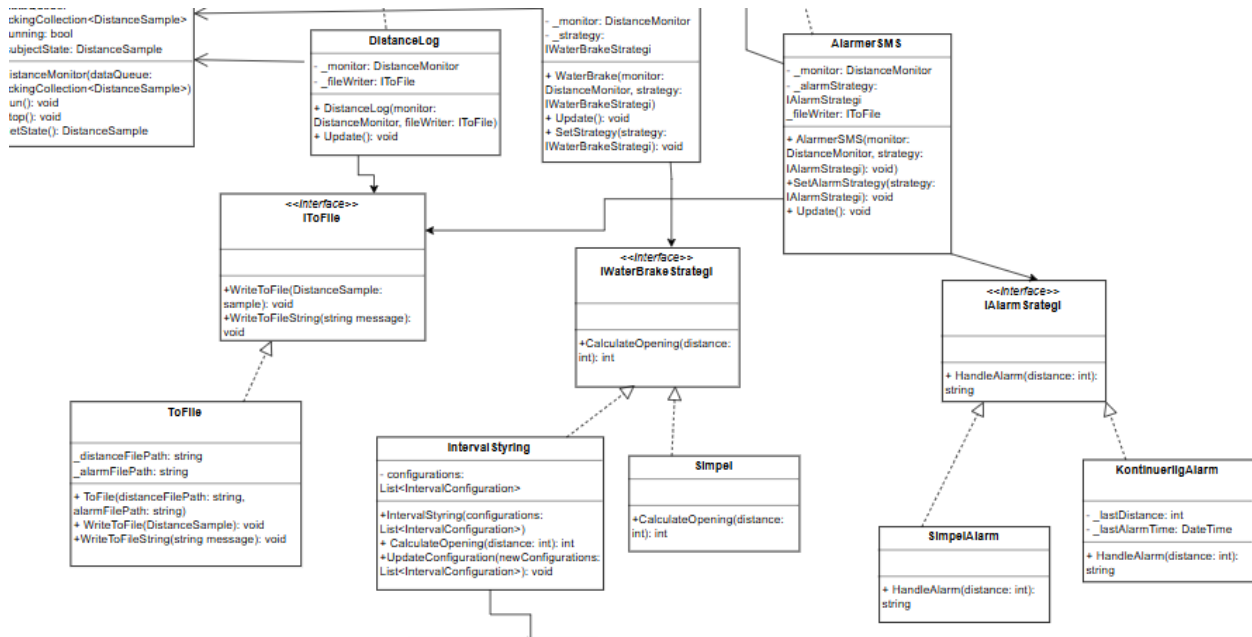
Figur 29 Konsoloutputtet for delopgave 22

I konsollen ser målingerne sådan ud, og de er ens i txt.-filen



Figur 30 Screenshot af txt-fil, hvor DistanceSample bliver nedskrevet

Delopgave 24



Figur 31 UML for delopgave 24

Der var flere måder at koble alarmbeskederne til at blive nedskrevet i en fil. Jeg valgte løsningen, hvor `AlarmSMS` kender til interfacet `IToFile`. Denne tilgang sikrer, at alt det, som `AlarmSMS` havde tænkt sig at udskrive til konsollen, også bliver skrevet ned i den ved at kalde metoden `WriteToFile`. For at understøtte dette opdaterede jeg de forskellige `IAlarmStrategi`-implementeringer, så de returnerer en string, der repræsenterer den besked, de ellers ville have udskrevet til konsollen. Denne string bliver derefter returneret til `AlarmSMS`, hvor den både bliver sendt til konsollen og logget i den ved hjælp af en ny metode, `WriteToFileString`, som jeg implementerede i `IToFile`, for ikke at få blandet data, har jeg lavet så den nedskriver i 2 forskellige filer, en til `DistanceSamples` og en til `Alarm` beskeder.

Delopgave 25

Her kan man se i konsollen at når jeg kører programmet og har sat distancen til at være 4, så starter den ud med at lave "ALARM", hvor den senere bliver til "ADVARSEL",

```
Tryk på følgende taster for at vælge en funktion:
'q' - Stop programmet
'i' - Skift til IntervalStyring
's' - Skift til SimpelStyring
'm' - Skift til SimpelAlarm
'k' - Skift til KontinuerligAlarm
'u' - Opdater konfiguration
1/14/2025 6:38:02 PM: Distance: 4, Sensor: 1
WaterBrake åbning: 15%
ALARM: Vandstanden er kritisk lav (4 cm)!
1/14/2025 6:38:12 PM: Distance: 4, Sensor: 1
WaterBrake åbning: 15%
ALARM: Vandstanden er kritisk lav (4 cm)!
1/14/2025 6:38:22 PM: Distance: 4, Sensor: 1
WaterBrake åbning: 15%
ALARM: Vandstanden er kritisk lav (4 cm)!
1/14/2025 6:38:32 PM: Distance: 2, Sensor: 1
WaterBrake åbning: 15%
ALARM: Vandstanden er kritisk lav (2 cm)!
1/14/2025 6:38:42 PM: Distance: 4, Sensor: 1
WaterBrake åbning: 15%
ALARM: Vandstanden er kritisk lav (4 cm)!
1/14/2025 6:38:52 PM: Distance: 4, Sensor: 1
WaterBrake åbning: 15%
ALARM: Vandstanden er kritisk lav (4 cm)!
1/14/2025 6:39:02 PM: Distance: 6, Sensor: 1
WaterBrake åbning: 15%
ADVARSEL: Vandstanden er lav (6 cm).
1/14/2025 6:39:12 PM: Distance: 8, Sensor: 1
WaterBrake åbning: 15%
ADVARSEL: Vandstanden er lav (8 cm).
```

Figur 32 Konsoloutput fra delopgave 24

Det samme bliver nedskrevet i filerne, hvor de er i hver deres fil.

```
1/14/2025 6:38:02 PM: ALARM: Vandstanden er kritisk lav (4 cm)!
1/14/2025 6:38:03 PM: ALARM: Vandstanden er kritisk lav (4 cm)!
1/14/2025 6:38:04 PM: ALARM: Vandstanden er kritisk lav (4 cm)!
1/14/2025 6:38:05 PM: ALARM: Vandstanden er kritisk lav (2 cm)!
1/14/2025 6:38:06 PM: ALARM: Vandstanden er kritisk lav (4 cm)!
1/14/2025 6:38:07 PM: ALARM: Vandstanden er kritisk lav (4 cm)!
1/14/2025 6:38:08 PM: ADVARSEL: Vandstanden er lav (6 cm).
1/14/2025 6:38:09 PM: ADVARSEL: Vandstanden er lav (8 cm).
```

Figur 33 Filen hvor Alarmbeskederne bliver nedskrevet.

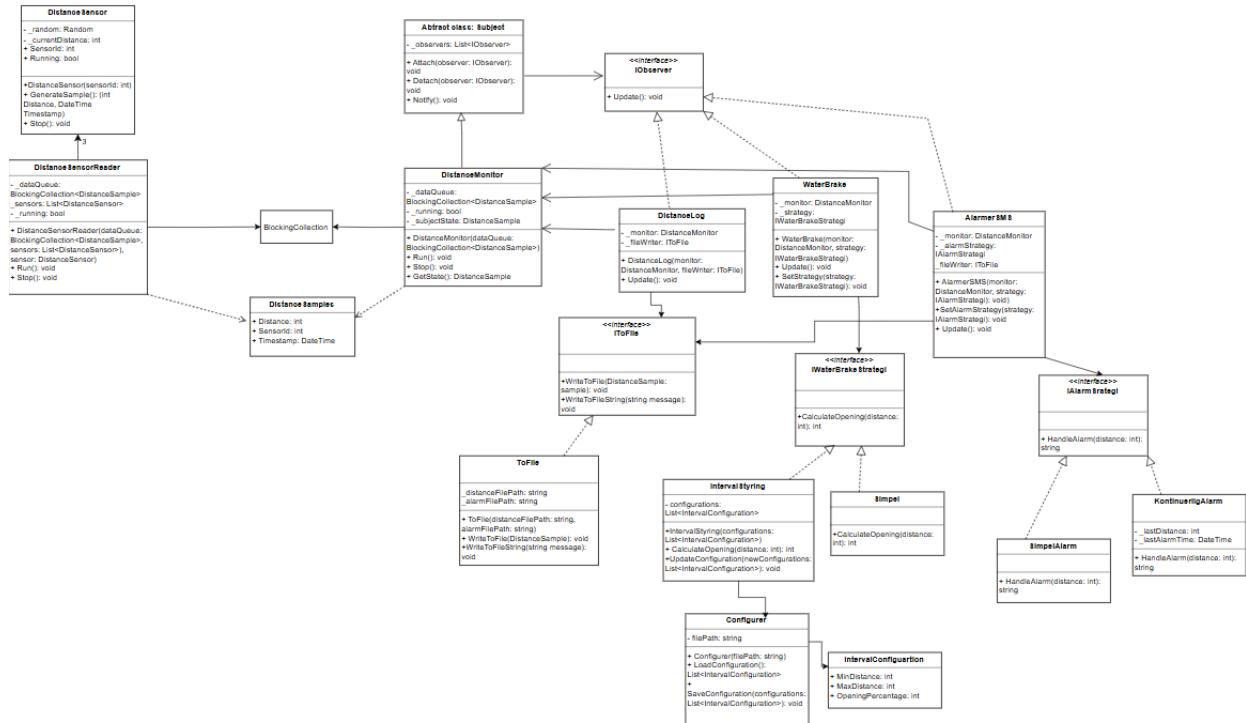
```
1/14/2025 6:38:02 PM: Distance: 4, Sensor: 1
1/14/2025 6:38:12 PM: Distance: 4, Sensor: 1
1/14/2025 6:38:22 PM: Distance: 4, Sensor: 1
1/14/2025 6:38:32 PM: Distance: 2, Sensor: 1
1/14/2025 6:38:42 PM: Distance: 4, Sensor: 1
1/14/2025 6:38:52 PM: Distance: 4, Sensor: 1
1/14/2025 6:39:02 PM: Distance: 6, Sensor: 1
1/14/2025 6:39:12 PM: Distance: 8, Sensor: 1
```

Figur 34 Filen hvor DistanceSamples bliver nedskrevet.

Delopgave 26

1.26 Delopgave 26

Medtag et samlet UML klassediagram i journalen for de opgaver du har løst.



Figur 35 UML for hele systemet

Systemet er også i mappen med zipfilen i png og svg.