AARHUS UNIVERSITET
INGENIØRHØJSKOLEN

EKSAMEN

Kursus:	ST3ITS3 – Programmering 3 - Hjemmeeksamen
Eksamensdato:	
Varighed:	**12** timer
Underviser:	Michael Sørensen Loft, Henrik B. Kirk
Eksamenstermin:	Vinter 2024

Praktiske informationer:

Opgavebesvarelsen skal afleveres i ZIP-format

Husk at uploade og aflevere i WISEflow. Du vil modtage en elektronisk afleveringskvittering, straks du har afleveret.

Husk at aflevere til tiden, da der ellers skal indsendes dispensationsansøgning.

Husk angivelse af navn og studienummer på alle sider, samt i dokumenttitel/filnavn.

Hjælpemidler:
Alle hjælpemidler må benyttes, herunder generativ AI og internettet som opslagsværktøj, men husk, at du selv skal kunne forklare alt det, du afleverer.
Husk at opgaven er en personlig opgave, så du må **ikke** lave løsningen sammen med andre.

Indledning

Denne tekst definerer eksamensopgaven i faget ST3ITS3, Programmering 3, for indeværende eksamenstermin på diplomingeniørstudiet i Sundhedsteknologi på ECE, Aarhus Universitet.

Eksamensopgaven ligger til grund for den mundtlige eksamen i faget. Med udgangspunkt i opgavens besvarelse vil du blive eksamineret i fagets indhold.

Opgavens besvarelse (”afleveringen”) skal bestå af netop 2 dele:

- En *journal* i PDF-format, hvor du besvarer de spørgsmål der er givet i de enkelte delopgaver.

- En *implementering* i en Microsoft Visual Studio-solution, som indeholder implementeringen af det system der efterspørges i opgaven.

Journalen skal afleveres som hoveddokumentet, implementeringen skal samles og afleveres i netop **en** samlet ZIP-fil. Filen skal være navngivet efter følgende format: `st3its3-v24-`*`studienummer-fuldeNavn`*`.zip`, hvor *`studienummer`* og *`fuldeNavn`* skal erstattes af studienummer hhv. fulde navn. Et eksempel på en korrekt navngivet fil er ”`st3its3-v24-20111223-HenrikBitschKirk.zip`”.

Formålet med opgaven er at give dig mulighed for at demonstrere din kunnen og viden inden for fagets læringsmål. Der lægges derfor vægt på at opgavens løsning demonstrerer din erhvervede viden inden for fagets forskellige emner.

Opgaven er inddelt i et antal delopgaver, hvor af nogle bygger videre på hinanden. Det fremgår af hver enkelt delopgave hvad der skal afleveres i journalen hhv. implementeringen af opgaven. Al implementering skal finde sted i den Microsoft Visual Studio solution der afleveres.

Hvis du går i stå i en opgave, bør du gå videre til en af de efterfølgende opgaver.

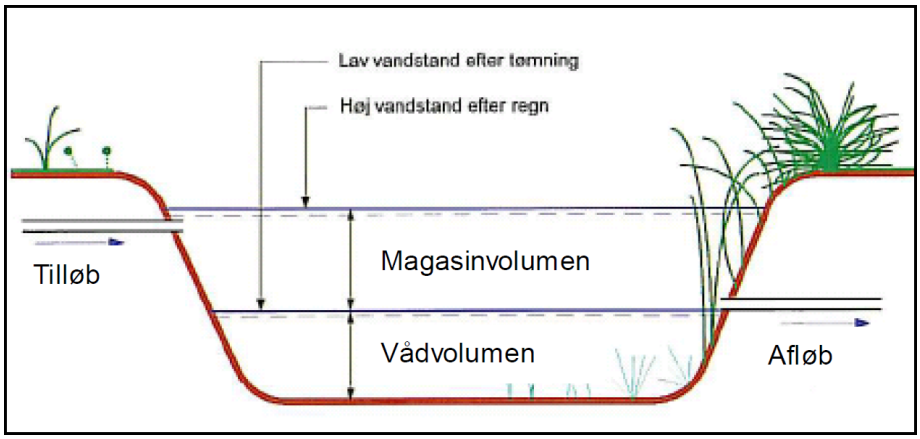
1 Overvågning og styring af regnvandsbassiner

Denne opgave omhandler et system til overvågning og styring af regnvandsbassiner.

Et regnvandsbassin er en kunstigt skabt sø, som har til funktion at opmagasinere og forsinke regnvand, når der falder meget regn. Vandet bliver ledt ind i bassinet gennem et tilløb og ud gennem et afløb. I afløbet sidder en vandbremse, som bruges til at styre, hvor hurtigt vandet forlader bassinet.

Bassinet har et vådvolumen, som er den mængde vand, der ligger under afløbets niveau. Det har også et magasinvolumen, som er den mængde vand, det kan indeholde i tillæg til dets vådvolumen før vandet når til den øverste kant af magasinet.

Figur 1 viser en illustration af et regnvandsbassin.



Figur 1 Regnvandsbassin[1]

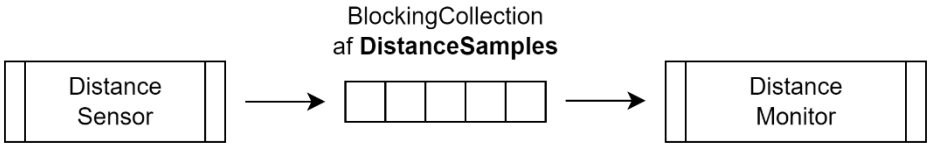
Systemet indeholder:

- En afstandssensor, som angiver, hvor mange cm under magasinets kant vandspejlet befinder sig.
- En vandbremse i afløbet, som kan åbnes mellem 0% og 100%, hvor 0 er helt lukket og 100% er helt åben. Vandbremsen kan også sættes til fx 20% åben eller 65% åben.

[1] Figuren er fra https://www.sdu.dk/da/forskning/forskningsformidling/citizenscience/campus-odense-active-living/regnvandsbassiner

1.1 Delopgave 1

En skitse af systemet (*ikke* et UML klassediagram) er givet herunder:



Figur 2: Systemskitse

Kravene til de tre klasser er som følger:

`DistanceSensor:`

- Skal køre i en selvstændig tråd.
- Skal indeholde en heltals-variabel, som repræsenterer den aktuelle afstand i cm fra kanten af bassinet til vandspejlet. Variablen skal initialiseres til at være 80 cm. Dette tal skal benyttes, når der efterfølgende laves en `DistanceSample`.
- Skal producere et tilfældigt heltal i intervallet 1-10 en gang hvert tiende sekund. Hvis tallet er mindre end 6, skal der trækkes 2 cm fra afstanden til vandspejlet (dvs. vandstanden stiger). Hvis tallet er 6 eller større, skal der lægges 2 cm til afstanden til vandspejlet (dvs. vandstanden falder).
- Skal sikre, at afstanden fra kanten af bassinet til vandspejlet ikke kan blive negativ.
- Skal generere og sende en `DistanceSample` til `DistanceMonitor` vha. den `BlockingCollection`, der forbinder `DistanceSensor` med `DistanceMonitor.`

`DistanceSample:`

- Skal indeholde afstanden.
- Skal indeholde sensor ID (et nummer du tildeler den enkelte sensor).
- Skal indeholde tidspunktet for målingen.

`DistanceMonitor:`

- Skal køre i en selvstændig tråd.
- Skal modtage Distance-samples fra `DistanceSensor` ved at tage dem fra den `BlockingCollection`, der forbinder `DistanceSensor` med `DistanceMonitor.`
- Skal behandle de modtagne Distance-Samples (målinger). Databehandlingen skal bestå i at udskrive de modtagne samples i konsollen, efterhånden som de bliver tilgængelige.

Systemet skal kunne lukkes pænt ned, ved et tryk på tasten 'q'.

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.

Redegør i din journal ligeledes for fordelene ved at benytte en `BlockingCollection` til kommunikationen mellem de to klasser.

1.2 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.

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(per) du mener overholdes bedre med denne adskillelse.

1.4 Delopgave 4

Udvid din implementering med de designændringer du har foreslået i delopgave 3. Vis i din journal konsoloutputtet fra en kørsel af dit program med de implementerede ændringer.

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.

1.6 Delopgave 6

Implementer de udvidelser du har designet i delopgave 5.

Vis et eksempel på konsoloutputtet efter en kørsel i din journal.

1.7 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 > afstand > 61	100
60 > afstand > 41	80
40 > afstand > 21	50
20 > afstand >= 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.

1.8 Delopgave 8

Implementer de udvidelser du har designet i delopgave 7.

Vis et eksempel på konsoloutputtet efter en kørsel i din journal, hvor du skifter mellem de to styringsmetoder imens programmet kører.

1.9 Delopgave 9

Skriv en unittest af din implementation af intervalstrategien fra Tabel 1.

Redegør i journalen for dine valg af testværdier og inkluder et screenshot af en kørsel af testen.

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.

1.11 Delopgave 11

Implementer dit design fra delopgave 11 og vis eksempler på konsoloutputtet efter to kørsler med forskellige konfigurationsværdier i din journal.

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 klassediagram.

1.13 Delopgave 13

Implementer dit design fra delopgave 12 og vis eksempler på konsoloutputtet efter en kørsel i din journal.

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 cm +54 cm) / 2 = 52 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.

1.15 Delopgave 15

Implementer dit design fra delopgave 14 og vis eksempler på konsoloutputtet efter en kørsel i din journal.

1.16 Delopgave 16

Skriv en unittest af din implementation af håndteringen af flere sensorer fra delopgave 14.

Redegør i journalen for dine valg af testværdier og inkluder et screenshot af en kørsel af testen.

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.

1.18 Delopgave 18

Implementer dit design fra delopgave 17 og vis eksempler på konsoloutputtet efter en kørsel i din journal.

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.

1.20 Delopgave 20

Implementer dit design fra delopgave 19 og vis eksempler på konsoloutputtet efter en kørsel i din journal.

1.21 Delopgave 21

Det ønskes, at de opsamlede data fra `DistanceSample's` bliver indsendt til en central server ved at benytte et REST API.

Redegør i din journal for hvordan dette krav vil kunne blive overholdt.

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 var 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.

1.23 Delopgave 23

Implementer de udvidelser du har designet i delopgave 22.

Vis et eksempel på logfilen efter en kørsel i din journal.

1.24 Delopgave 24

Log'en skal udvides, så advarsler og alarmer også bliver logget i samme fil, som afstandssmålinger.

Redegør for de nødvendige udvidelser og/eller ændringer til dit system. Redegørelsen skal være i form af et UML klassediagram.

1.25 Delopgave 25

Implementer dit design fra delopgave 24. Vis i journalen et konsoloutput fra en kørsel af dit program.

1.26 Delopgave 26

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