Oppgavesettet består av 2 sider

HØYSKOLEN KRISTIANIA
PG3302 Software design
Mappeeksamen

Gjennomføring

- **Mappeeksamen, skal leveres i gruppe** (2-5 personer).
- **Arbeidsmapper** kan vises foreleser i øvingstimer for **tilbakemelding underveis**.
- **Presentasjonsmappe** skal leveres via **WISEflow.**
- Den endelige presentasjonsmappen skal inneholde:
    1. Dokument som beskriver prosessen og innhold/bruk av løsningen (pdf)
    2. Kildekode og evt. andre nødvendige filer i den sammenheng (zip)
    3. *Om mulig:* Kjørbar (.exe-)versjon, *som kan startes via Windows* (zip)

Leveranseinfo, 3 deler

Her følger litt mer info om de 3 delene i leveransen:

1. Dokument som beskriver prosessen og innhold/bruk av løsningen (pdf)

Denne delen går ut på å lage en kort, skriftlig rapport som dokumenterer prosjektet dere har jobbet på. Foreslått lengde: minimum 1500 til 2000 ord (3 til 4 sider med vanlig størrelse og linjeavstand).

Elementer som bør være med:

- **Eksamensprosessen**: Skriv om hvordan dere har jobbet med besvarelsen.
- **Løsning, utfordringer & fancy features:** Si hva dere lagde og hvorfor (inklusive DB- eller filformat og hvorfor dette.) Litt om hva dere slet med, og hva dere er spesielt fornøyd med.
- **Liste med innhold fra pensum:** En kort liste med hvilke elementer fra pensum dere har med, og hvor/hvordan det er brukt.
- **Beskrivelse av temaer som ikke sees i koden:** Refactoring, parprogrammering, SOLID?, etc.
- **Kjente bugs:** Kjenner dere til noen feil i programmet? Nevn de i så fall her.
- **Kildehenvisning, VIKTIG:** Om det oppdages at dere har benyttet ferdig materiell som dere *ikke* opplyser om (ikke nevner her), kan det regnes som juks! Husk derfor å oppgi kilder, vi ønsker å unngå plagiatsaker!

Husk: Software design handler vel så mye om utviklingen av et program som det ferdige programmet. Pass på at dere dokumenterer både løsningen deres og prosessen godt!

2. Kildekode og evt. andre nødvendige filer i den sammenheng (zip)

Dette er kodeimplementasjonen av programmet. Lever kildekoden (og tilhørende filer), slik at sensor kan se programmeringen og kodestrukturen! *Merk:* Bør kunne åpnes i Visual Studio 2022.

Husk: Vi har lært om modellering av softwareløsninger i dette emnet, gjennom klasse- og sekvensdiagrammer i UML. Figurer og annet innhold dere produserer i den sammenheng bør også leveres her. Det kan være screenshots fra digitale verktøy, mobilfoto av papirbaserte tegninger, etc.

3. *Om mulig:* Kjørbar (.exe-)versjon, *som kan startes via Windows* (zip)

Kjørbar versjon skal, om mulig, kunne benyttes på Windows 11. Sørg for at både .exe-fil og eventuelle nødvendige mapper blir med. Har dere ikke mulighet til å bygge en versjon som kjører på

Windows, forklar hvorfor og vær ekstra nøye på at innholdet i pkt. 2 (over) er dekkende, slik at sensor kan bygge kjørbar versjon selv.

Utforming av løsning

Denne eksamenen går ut på å **designe en softwareløsning**, deretter **programmere softwareløsningen i C#**.

Designe en softwareløsning

Når det gjelder "designe en softwareløsning", menes det med fokus på kodestrukturen. brukeropplevelse, ikke visuell stil, ikke markedsaspektet eller annet som ikke hører inn under pensum i emnet.

Programmere løsningen i C#

Dere står fritt til å velge tema/domene programmet skal håndtere, så lenge løsningen dekker innholdet for standard 3-lags arkitektur:

- Datalagring i bunn (lagring til DB, JSON-fil eller XML-fil).
- Domenelogikk i midten.
- Console-basert frontend. (*Ikke* mye fokus på UI-opplevelsen, det er ikke del av pensum. Men ryddig frontend kode er bra.)

Det er forventet at løsningen inneholder minst 2 prosjekter:

- C#, Console App
- C#, NUnit Test Project

Pass på at dere får med et godt utvalg elementer fra pensum! (Se avsnittet under.)

Pensum, hva bør være med

Elementer fra pensum: (dette er ikke en komplett liste, vurderer din gruppe å benytte flere/andre elementer fra pensum er det helt fint)

- UML diagrammer.
- C# .NET, med en del finesser fra språket: properties!, evt. operator overloading, …
- SOLID (beskriv gjennom dokumentasjonen hvilke prinsipper dere har fokus på).
- Design patterns (plukk noen få av de vi har lært om: *prøver man å benytte alle blir det kaos*).
- Refactoring (sørg for at koden holdes oversiktlig og lett å sette seg inn i). Beskriv refactoring prosessen deres, den kan være vanskelig å se ut fra koden.
- Lagdeling
- Unit testing. *Merk:* Det er omfattende å skrive nok unit tests til å dekke en hel løsning. Derfor ok om man bare benytter det på *deler* av løsningen. Men beskriv i så fall hva dere velger å teste og hvorfor. Implementer tester slik at dere får vist bredden av hva dere kan om unit testing.
- Multithreading / event-basert kode (pass i så fall på at dere skriver trådsikker kode).
- Parprogrammering. Beskriv i så fall hvordan dere benytter dette, samt erfaringene dere har gjort dere.
- Bruk av versjonskontroll.

Merk: Tenk god struktur, ikke nødvendigvis hva som hadde vært praktisk for et så lite prosjekt utenfor skolesammenheng. Få med mange elementer fra pensum. Gjør dere noe som ikke virker logisk (fordi dere vil vise at dere behersker en gitt del av pensum, men finner ikke en perfekt plass å vise den på), forklar hvorfor dere har det med, *både* som kodekommentar og i deres vedlagte pdf dokument.

- Slutt på oppgavesettet -