

# Objektorienteret Programmering 1

# Workshop 1 af 2



# Hvad er blevet gennemgået i starten af September?

- Variable
- Løkker
- Branching
- Arrays
- Lidt om objektorienteret analyse og design
- Klasser
- Objekter
- Constructors
- Objekters tilstande og accesor & mutator metoder
- Metoder
- Pakker
- Access modifiers
- ArrayList
- Maps
- Søgning
- Sortering
- Enums
- Switch
- Arv
- Polymorfi
- GUI
- Event handling
- Interfaces
- Lidt om softwarearkitekturer
- Generic programming
- Java Collection Framework
- Exception handling



- Vi har udfordret jer
- I har været gennem meget materiale
- Hvis I nogenlunde kan tingene på sidste slide, så er I fornuftigt rustet til jeres fremtidige uddannelse (og eksamen i januar)
- Der bør være rigeligt for jer at give jer i kast med i juleferien



- Praktisk workshop i (næsten) alt I hvad semestret har budt på
- Fokus på programmering og forståelsen
- Praktisk opgave der kræver og giver forståelse for kursets emner
- Procedure:
  - Udvikling/implementering
  - Fælles statusmøde
  - Udvikling/implementering
  - Fælles statusmøde
  - osv. osv.
- Op med hånden hvis I går i stå
- Sæt jer sammen tre og tre
  - Sørg for at diskutere løsningerne igennem
  - Opvej for og imod




# Casebeskrivelse

- Moderne bygninger er for en stor dels vedkommende udstyret med sensorer, der muliggør overvågning af blandt andet temperatur og luftkvalitet.
- I skal udvikle et bygningssystem, der kan opsamle målinger fra sådanne bygninger.
- Bygningerne er kendetegnet ved at have et varierende antal sensorer til temperatur målinger (i grader celsius) og måling af CO2 niveau (i ppm<sup>2</sup> (parts per million i anden)).
- Det skal være muligt, at oprette nye bygninger i systemet ved at angive deres navn, deres placering (geografisk/adresse) og de sensorer, som de indeholder.
- Systemet skal kunne opsamle målinger fra flere bygninger, og opsamle målinger fra alle sensorer i hver bygning. "Opsamlingen" sker indtil videre ved, at en person indtaster værdier fra målingerne, men tænkes på sigt at foregå automatisk.
- Systemet skal give et overblik over, hvilke bygninger, der eksisterer i systemet og muliggøre, at man kan se en historisk oversigt (graf) over, hvordan miljøet i en given bygning udvikler sig.



# Overordnet tilgang

- Der lægges i opgaven op til, at der skal udvikles et program, der har en grafisk brugergrænseflade
  - Del jeres design op således, at I har en “forretningslogik”-del og en “brugergrænseflade”-del
  - Start med at få forretningslogikken til at virke
  - “Skjul” forretningslogikken bag et interface
  - Tilføj brugergrænseflade senere
- 




# SÅ GÅR VI I GANG



# Sprint 1

# Objekt-orienteret analyse (20 min/5-10 min til status)

- I skal nu identificere klasser, metoder og attributter i jeres forretningslogik
  - Lav efterfølgende et UML diagram, der viser dokumenterer jeres forretningslogik.
  - Mål efter sprint 1:
    - Et færdigt udkast til et system design
    - Et UML diagram der viser jeres forretningslogik i grove træk
      - De vigtigste klasser
      - Centrale metoder
- 





# Sprint 2

# Realisering af design (40 min/10 min til status)

- Lav en implementering af jeres forretningslogik
  - Klasser
  - Metoder
  - Attributter
  - Constructors
- Revidér løbende jeres design efter behov
  - Ændringer skal naturligvis kunne forklares
- Mål efter sprint to
  - En prototype
  - UML diagram, revideret til at afspejle ændringer og med flere detaljer (hvis I har tid)
    - Alle metoder
    - Argumenter til metoder
    - Attributter
    - Access Modifiers



# Sprint 3 (30 min/5-10 min til status)

- Test af forretningslogikken
  - Tilføj/fjern bygninger af forskellige arter
  - Vis oversigt over bygninger (tekstuel)
  - Tilføj målinger for en bygning
  - Udskriv serie af målinger for bygninger
- Debugging
- Tilrettelser
- Mål efter sprint 3:
  - Et system, der lever op til specifikationerne på forretningslogik-siden



## Sprint 4 (15 min/5 min til status)

- Lav “mock-up” af en grafisk brugergrænseflade
  - Lav tegninger, der viser en grov skitse af brugergrænsefladen
  - Lav en tegning for hver særskilt gruppering af funktionalitet
    - Eks. “Bygningsadministration”, “Målingsoversigt”
  - Husk det eksplicite krav om graf til visualisering af målingshistorik.
  - (Hvis I er hurtigere til at tegne på computeren, er det helt fint)
- Mål efter sprint 4:
  - Mockups for brugergrænsefladen, der dækker al den forespurgte funktionalitet.




## Sprint 5 (25 min/5 min til status)

- Lav et nyt JavaFX-projekt
- Design jeres brugergrænseflade (eksempelvis ved brug af TabPane)
- Mål efter sprint 5:
  - Et grafisk brugerinterface (endnu uden funktionalitet)



# Sprint 6 (40 min/10 min til status)

- Lav en instans af jeres forretningslogik fra jeres grafiske brugergrænseflade
    - Benyt initialize()-metoden hertil
  - Lav event-handlers for jeres controls, der interagerer med forretningslogikken
    - Fokusér i denne sprint på, at der kan tilføjes data til jeres back-end
      - Oprette bygninger
      - Oprette målinger
  - Mål efter sprint 6
    - Kontakt til forretningslogikken fra jeres brugergrænseflade
    - Mulighed for at indtaste data
- 



# Sprint 7 (45 min/10 min til status)

- Visualisering af data i jeres brugergrænseflade
  - Automatisk opdatering af lister, felter
  - Brug af graf til at vise målingshistorik
    - <http://docs.oracle.com/javafx/2/charts/line-chart.htm#CIHGBCFI>
- Mål efter sprint 7
  - Udkast til færdigt system



# NÆSTE GANG



# Næste lektion (fredag d. 9. 12.)

- Fortsætter vi hvor vi slap i dag
  - GUI
- Opgaveformuleringen udvides med yderligere dele

