# Architecture and GUI Intro

# Øvelse 1: Terminologi

Nu skal du teste din forforståelse af dagens emne(r) med udgangspunkt i dagens forberedelse.

Del teamet op i to mindre grupper, og brug **Møde på midten** (beskrivelse følger i punktform) i hver gruppe til at diskutere begreberne ”Package Diagrams”, ”Logical Architecture”, ”Logical Architecture Refinement” og ”Grafisk brugergrænseflade (en: Graphical User Interface (GUI))”. Husk, at alle i gruppen skal have taletid.

*Tidsramme: 20 minutter*

# Øvelse 2: Planlæg lagdeling og GUI

I denne øvelse skal du genoverveje arkitekturen for din Disaheim projekt. Uden ordentlig planlægning af arkitekturen, og dermed organisering af koden i løsningen, kan man nemt løbe ud i problemer, såsom at koden til håndtering af menuen, styring af forretningsobjekter og deres logik samt kontakt til persistens-koden i værste tilfælde er blandet helt sammen uden nogen form for lagdeling. Dette kan i værste tilfælde give en tætkoblet og fastlåst kode, som er svær at vedligeholde og teste.

Du skal derfor i denne øvelse planlægge arkitekturen i Disaheim projekt, hvor du nu konkret benytter følgende tre lag:

* **UI-laget** indeholder al kode, som har med brugergrænsefladen at gøre
* **Applikationslaget** formidler kommunikationen mellem UI-laget og Domain-laget
* **Domain-laget** indeholder modellen med forretningsobjekterne (”domain objects”) og deres forretningslogik

Der stilles følgende krav til hvert af de tre lag:

* UI-laget:
  + Al kode, som skriver til eller læser fra brugergrænsefladen samt håndterer brugerens valg, skal være i dette **og kun i dette lag**
  + Det eneste andet lag, som UI-laget kender og har kontakt til, er applikationslaget
* Applikationslaget:
  + Dette lag skal indeholde en Controller, som håndterer kommunikationen mellem UI-laget og Domain-laget
  + Det eneste andet lag, applikationslaget kender og har kontakt til, er Domain-laget. Bemærk at applikationslaget **intet kendskab har til UI-laget**
* Domain-laget:
  + Dette lag håndterer alle forretningsobjekter i løsningen og **har intet kendskab til UI- eller applikationslaget**

## Øvelse 2.1: Repræsentation af lag

Der er flere forskellige måder at repræsentere de forskellige arkitekturlag på (både fysisk og logisk) i C#-koden og i Visual Studio-løsningen.

* Overvej, hvilke forskellige måder du konkret kan repræsentere 3-lags arkitekturen for din C#-løsning i Visual Studio, samt fordele og ulemper ved disse. Inddrag de principper i SOLID og GRASP, du kender på nuværende tidspunkt.
* Overvej, hvilken snitflade hvert arkitekturlag i dit valg af lagdelingen skal tilbyde i løsningen og hvilke klasser(objekter), der er i hvert lag, og hvilke af disse der udbydes af lagets snitflade.
* **Udarbejd et pakkediagram**, der visuelt illustrerer din lagdeling og logiske arkitektur med indbyrdes afhængigheder mellem lag og pakker. Bemærk, at et pakkediagram typisk kun viser titlen på klasser, og ikke klassernes attributter og operationer. Relationer mellem klasser skal vise, hvorledes klasser er afhængige af hinanden (en pil med stiplet linje i UML), f.eks. at en klasse bruger (en: ’uses’) en anden klasse. Se for inspiration nedenunder på et uddrag af en lagdelt DCD med en Controller (dog kun med fokus på Book) fra øvelse 3 i uge 16.

Et billede, der indeholder tekst, kvittering, linje/række, diagram

Automatisk genereret beskrivelse

## Øvelse 2.2: Den grafiske brugergrænseflade (GUI = Graphical User Interface)

Et moderne UI-design vil oftest være en grafisk brugergrænseflade. Derfor skal du nu skitsere en mulig GUI for eftermiddagsprojekt-løsningen.

Ud fra din erfaring med brugergrænseflader i Windows (eller et andet operativ­system), gør følgende:

* Overvej først, hvilke grafiske komponenter, der kunne indgå i løsningen (knapper, tekstbokse, lister, mm.)

En konsolapplikations UI-del repræsenteres af kun én simpel tekstbaseret UI-kontrol udelukkende med mulighed for at skrive til og læse fra konsolvinduet. En GUI består af mange forskellige UI-kontroller, som grafisk har meget forskellige funktioner lige fra knapper, menuer, ikoner, drop-down-lister til forskellige former grafisk visning af træstrukturer, lister, billeder, tekster, mm.

* Tænk helt frit ud af boksen, brug din fantasi, og lav et grafisk wireframe/skitse for Disaheim projekt (gerne flere alternativer)

# Øvelse 3: Repetition/Færdiggørelse af tidligere opgaver

Resten af dagen er afsat til repetition. Det vil sige, at du ikke vil blive introduceret til noget nyt på denne dag, men du kan vælge selv, hvordan du vil bruge dagen, hvor du har brug for det:

1. Gå tilbage til de øvelser i tidligere opgaver, du ikke har fået løst endnu, og gøre dem færdig; du vælger selv.
2. Genløs alle programmeringsøvelser i tidligere opgaver med den indsigt, du har nu.
3. Hvis du har løst alle opgaver, så gennemgå og revurdér dem og se, om der kunne være behov for at tilpasse koden; f.eks. gøre den mere effektiv, kortere, mere læselig eller bedre til at håndtere fejlsituationer, mm.
4. Eller arbejd på det projekt, du har behov for lige nu (f.eks. projektarbejdet)
5. Husk at skrive kommentarer til hvad programmet gør.

Anvend den samarbejdsform, der passer bedst til din aktivitet individuelt eller i samarbejde med dit team.