# Architecture and GUI Intro

|  |  |
| --- | --- |
| **Læringsmål** | Du kan:   * **1Pf1:** anvende centrale metoder til at specificere og konstruere algoritmer [...] * **1Pf2**: anvende centrale faciliteter i programmeringssproget til realisering af algoritmer, designmønstre, abstrakte datatyper, datastrukturer, designmodeller og brugergrænseflader * **1Pf3**: anvende et i professionen udbredt, integreret udviklingsværktøj […] til at designe og konstruere praksisnære applikationer baseret på en lagdelt arkitektur * **1Pk3**: i en struktureret sammenhæng tilegne sig ny viden, færdigheder og kompetencer inden for programmeringssprog, udviklingsværktøjer, programmeringsteknikker og programdesign |
| **Forventet læringsudbytte [SOLO]** | Designsporet:   * [Unistrukturel] Du kan genkende enkelte kendetegn ved **strukturmodellering (en: modeling structure)** – designklassediagram (DCD) og pakkediagram (PD) * [Unistrukturel] Du kan genkende enkelte kendetegn ved **adfærdsmodellering (en: modeling behaviour)** – sekvensdiagram (SD)   Programmeringssporet:   * [Multistrukturel] Du kan beskrive flere kendetegn ved styring af **program flow** * [Multistrukturel] Du kan beskrive flere kendetegn ved **datahåndtering** * [Multistrukturel] Du kan beskrive flere kendetegn ved simple **C# datatyper** * [Multistrukturel] Du kan beskrive flere kendetegn ved **C# klasse (objekt)** * [Multistrukturel] Du kan beskrive flere kendetegn ved **Programstruktur** * [Unistrukturel] Du kan identificere enkelte kendetegn ved **Applikationstyper** – konsolapplikation, bibliotek (en: Class Library), GUI (Graphical User Interface) applikation med enkelte grafiske GUI-komponenter |
| **Din forberedelse** | Designsporet:   * [Larman]: Kapitel 13 Logical Architecture and UML Package Diagrams (fokusér på pakkediagrammer og lagdeling) * [Larman]: Kapitel 34 Logical Architecture Refinement   Programmeringssporet:   * Se på Tusindfryd analyse-artefakterne *igen igen* (OM, DM, UC, SSD, OC) med fokus på ”Gennemse produktion” * Se på jeres tidligere udarbejdede landkort fra systemudvikling |

I denne opgave skal du skal du arbejde med it-arkitekturen for et it-system ved at gå mere i dybden med den logiske arkitektur samt UML pakkediagrammet. Derudover introduceres du til grafiske brugergrænse­flader som opstart til at du selv skal udvikle Windows GUI-applikationer med WPF i den følgende opgaver.

# Dagens citat:

In many cases the user interface to a program is the most important part for a commercial company: whether the programs works correctly or not seems to be secondary.  
([Linus Torvalds](https://en.wikipedia.org/wiki/Linus_Torvalds))

# 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”. 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 tidligere HYDAC-løsning fra projektet inden efterårsferien. 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 din tidligere HYDAC-løsning, 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 HYDAC 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.

## 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 HYDAC-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 HYDAC-løsningen (gerne flere alternativer)

# Øvelse 3: Datamatikerens landkort

I denne øvelse skal du og dit team opdatere teamets landkort og tilføje den viden, du har fået om designmodel­lerne SD og DCD. Det er vigtigt at beskrive artefakternes sammenhæng til de øvrige artefakter. Vær særligt opmærksom på, hvordan de forskellige artefakter påvirker hinanden, og hvordan du kan vise dette på en god måde.

Resultatet af denne øvelse burde være, at du og dit team har et landkort, der beskriver sammenhængen mellem følgende artefakter:

* Krav
  + Use cases (UC)
  + Systemsekvensdiagram (SSD)
  + Operationskontrakt (OC)
* Forretningsmodellering
  + Objektmodel (OM)
  + Domænemodel (DM)
* Design
* Sekvensdiagram (SD)
* Designklassediagram (DCD)
* Pakkediagram (PD)

Du burde desuden også have en liste af kvalitetskriterier til de fleste af disse artefakter.

**Husk:** Det er tilladt at benytte farver. Alle midler gælder i henhold til at gøre det mere tydeligt, hvordan artefakterne hænger sammen.

# Øvelse 4: Tusindfryd

Arbejd videre med Tusindfryd-casen enten fra, hvor du er nået til i de tidligere opgaver eller fortsæt med at realisere nedenstående OC.

## Øvelse 4.1: SD ud fra OC

Du og dit team skal nu udforme sekvensdiagrammer (SD) samt tilpasse DCD’et ud fra operationerne vedrørende systemsekvensdiagrammet *Gennemse produktion*; hertil har du fået udleveret følgende 3 operationskontrakter (OC):

|  |  |
| --- | --- |
| **Operation:** | gennemsetAfMedarbejder(initialer:string) |
| **Cross References:** | Gennemse produktion |
| **Preconditions:** | * Medarbejder-objekt findes |
| **Postconditions:** |  |

|  |  |
| --- | --- |
| **Operation:** | vælgeProduktionsbakke(navn:string) |
| **Cross References:** | Gennemse produktion |
| **Preconditions:** | * produktionsbakke findes med et tilknyttet produktionsobjekt |
| **Postconditions:** | * ingen |

|  |  |
| --- | --- |
| **Operation:** | angivOptællingsinformation(antal:integer, dato:DateTime) |
| **Cross References:** | Gennemse produktion |
| **Preconditions:** | * produktionsobjekt findes * medarbejder-objekt er valgt |
| **Postconditions:** | * et optællingsobjekt blev oprettet med antal og dato og beregnetSlutantal og afvigelseIProcent (instance creation) * optællingsobjektet blev associeret til produktionsobjektet (association formed) * optællingsobjekt blev associeret til medarbejder-objekt (association formed) |

Sørg for, at dine kvalitetskriterier for SD og DCD bliver overholdt.

## Øvelse 4.2: Kvalitetssikring af SD’er

Når du og dit team er færdige med at udforme jeres SD’er, skal I have kvalitetssikret jeres artefakt med minimum et andet team. Lav rettelser til jeres artefakt og kvalitetskriterier ud fra behov.

## Øvelse 4.3: Implementering af kode ud fra SD

Fortsæt, hvor du og dit team slap sidst. Hvis I blev færdige med implementeringen af det forrige SD, så skal I fortsætte med SD’erne fra i dag. Det er tilladt at oprette ’fake data’.

**Bemærk:** Du og din sidemakker må gerne spørge dine medstuderende/os undervisere/lektiecafé-hjælperne til råds, hvis I er gået i stå.

# Øvelse 5: Opsummeringsøvelse (kl. 11.00)

Du og dit team skal lave en brainstorm over, hvordan I tænker, at vi bedst kan bruge de sidste 20 minutter af formiddagen til, dvs. opsummeringen. Send alle de ideer, I kommer på i teamet i en mail til både Jan ([jbro@ucl.dk](mailto:jbro@ucl.dk)) og Leif ([ldki@ucl.dk](mailto:ldki@ucl.dk)).

Den eneste betingelse vi undervisere stiller er, at I ikke må være ‘passive’ i opsummeringen, men deltage aktivt i en dialog. Så et forslag om, at vi undervisere skal præsentere vores facit for jer for dagens øvelse, behøver I ikke at skrive J Men ellers er vi undervisere meget lydhøre over for jeres ideer til at gøre opsummeringen spændende, underholdende og værdifuld for alle.