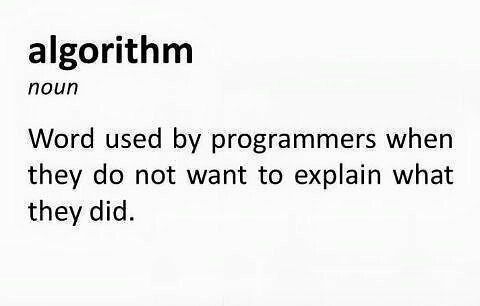
# Design Modeling with SD and DCD

|  |  |
| --- | --- |
| **Læringsmål** | Du kan:   * **1Pf1**: anvende centrale metoder til at specificere og konstruere algoritmer [...] |
| **Forventede  produkter** | * Udarbejdelse af pseudokode fra DCD’er og SD’er |
| **Forventet læringsudbytte [SOLO]** | Designsporet:   * [Unistrukturel] Du kan genkende enkelte kendetegn ved **strukturmodellering (en: modeling structure)** – designklassediagram (DCD)  [Unistrukturel] Du kan genkende enkelte kendetegn ved **adfærdsmodellering (en: modeling behaviour)** – sekvensdiagram (SD) |
| **Din forberedelse** | Designsporet:   * [Larman]:   + Kapitel 15 ”UML Interaction Diagrams”: 15.1 - 15.4 (pp. 221 – 238) *Hav fokus på hvordan et sekvensdiagram (SD) skal aflæses*   + Kapitel 16 ”UML Class Diagrams”: afsnit 16.1 til 16.8 (pp. 249 – 259) *Hav fokus på hvordan et designklassediagram (DCD) skal aflæses* * [Sequence and communication diagram](https://www.linkedin.com/learning/software-design-modeling-with-uml/sequence-and-communication-diagram) (video: kun til 7:08, se bort fra resten af video) * [Pseudokode kan være det sværeste programmeringssprog at mestre](https://www.version2.dk/artikel/pseudokode-kan-vaere-svaereste-programmeringssprog-at-mestre-1076140) |

Formålet med dagens opgave er, at du lærer at tolke designklassediagrammer (DCD) og sekvensdiagram­mer (SD), dvs. du skal udelukkende have fokus på at kunne læse og forstå DCD’er og SD’er og ikke på selv at lave dem. Det at lave dem, kommer vi til senere. Øvelserne i opgaven vil arbejde med nogle af sammen­hængene mellem et DCD og SD, og hvorledes et DCD og SD kan fortolkes til kode i form af pseudokode.

# Dagens ord:



Men det kommer du alligevel til ☺ med sekvensdiagrammer.

# Øvelse 1: Terminologi

Inden du skal i gang med øvelserne, skal du have testet din forforståelse af et sekvensdiagram og dets opbygning.

Del teamet op i mindre grupper (af 2 personer), og brug CL-strukturen **tænk-par-del** (se nedenstående fremgangsmåde) til at reflektere over begreberne:

1. *Sekvensdiagram: formål*
2. *Objektangivelse med livslinje (en: lifeline) og aktiveringssøjler (en: activation bars)*
3. *Besked (en: message) mellem objekter:* 
   * *syntaks med beskednavn, parametre, returværdi*
   * *pil med fuldt optrukken eller stiplet linje*
4. *Fragmenter: Opt, Alt og Loop (Opt = optional, Alt = alternative)*

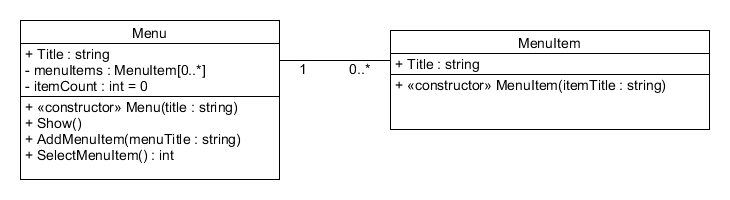
**Fremgangsmåde:**

1. Team: Del punkterne ud mellem jer, et punkt til hver gruppe af 2 personer (1 minut)
2. Individuelt: Skriv dine overvejelser ned om begreberne (5 minutter)
3. Par: Del jeres tanker med hinanden (2 minutter)
4. Par: Forbered jer på, hvad der skal deles med teamet (2 minutter)
5. Team: Præsentere på skift det, som I er blevet enige om, til hele teamet (ordet rundt) (2 minutter per gruppe)
   1. De øvrige omkring bordet tager noter og stiller spørgsmål

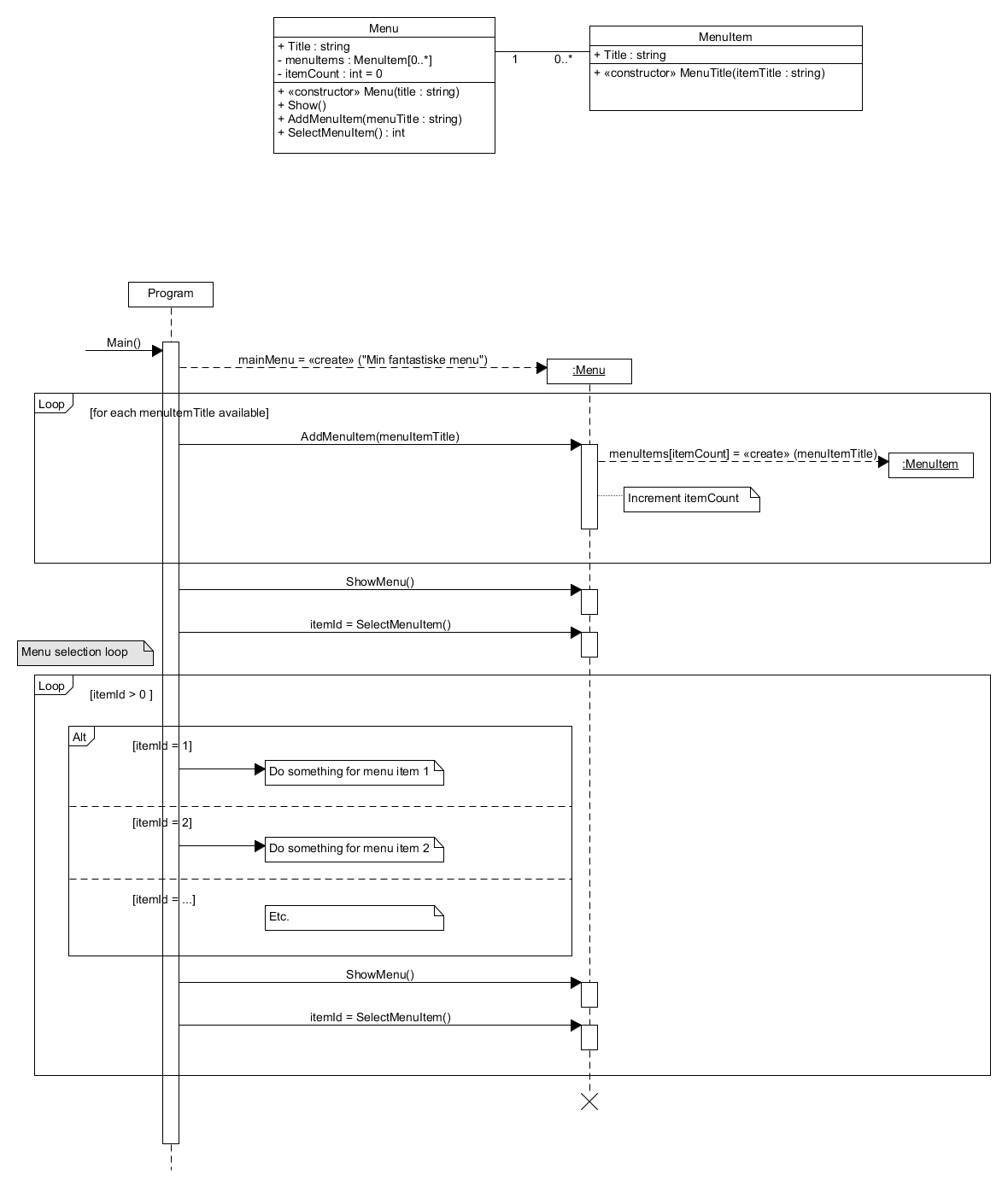
*Tidsramme: 20 minutter*

# Øvelse 2: Sammenhænge mellem DCD og SD

I fredagsopgaven Ex07-Menu i forrige uge, arbejdede du med følgende DCD (øvelse 2.5):



Dette DCD suppleres nu med følgende SD:



Bemærk, at et SD altid er på et højere abstraktionsniveau end konkret kode. Egentlig skal du ikke kunne se hvilket programmeringssprog, der anvendes, selvom objekt-orienterede programmeringssprog nok er de mest oplagte. Derudover er det fint, hvis SD’et også giver programmøren en vis frihed til at implementere den dynamiske adfærd, som vedkommende finder mest optimalt. Derfor er SD’et lidt mere overordnet, men dog alligevel specifik på de objekter, der interagerer. Med andre ord, den dynamiske del af et it-system illustreres, så der bibringes en algoritmisk forståelse for interaktionen mellem de viste objekter.

## Øvelse 2.1: Inspicér SD’et

Udfør følgende med **tænk par-del**:

* **Tænk**: Inspicér udelukkende SD’et foroven, og prøv at forstå, hvad de enkelte notationer betyder, og hvad hele SD’et prøver at illustrere.
* **Par**: Del dine observationer med din sidemakker samt hvad I vil dele med hele teamet
* **Del**: Benyt ordet rundt, og præsentér jeres fund. Prøv sammen i teamet at opnå en fælles forståelse.

## Øvelse 2.2: Find sammenhænge

Der er en sammenhæng mellem de to viste diagrammer, DCD’et og SD’et. Du skal nu øve dig i at læse og tolke denne sammenhæng.

Igen brug **tænk-par-del** til følgende:

* **Tænk**: Inspicér individuelt både DCD’et og SD’et, og find relaterede elementer mellem dem
* **Par**: Overvej med din sidemakker de fundne sammenhænge
* **Del**: Del dine fund med hele teamet (ordet rundt)

## Øvelse 2.3: Opstil kvalitetskriterier

For at man reelt kan se en sammenhæng mellem et DCD og SD, bliver man nødt til at være konsistent på tværs af de to diagrammer. Når du selv skal til at designe DCD’er og SD’er, er det vigtigt at blive guidet af nogle kvalitetskriterier, der sikrer sporbarhed mellem artefakterne.

Brug **tænk-par-del** til følgende:

* **Tænk**: Opstil nogle generelle kvalitetskriterier, som du finder vigtige for at sammenhængen mellem et DCD og SD er tydelig og for at undgå misforståelser.
* **Par**: Tal med din sidemakker de fundne sammenhænge og formulering af kvalitetskriterier
* **Del**: Del jeres fund med hele teamet, og lav en fælles liste, som du kan bruge fremover (du vil supplere til disse kriterier senere hen).

# Øvelse 3: Fra DCD og SD til pseudokode

Får man udleveret et DCD og et SD fra et designteam, kan det være en rigtig god idé i implementerings­teamet at skitsere pseudokode for at komme nærmere på den konkrete kode, og spotte nogle problemer, inden man reelt koder i et programmeringssprog. Brug af papir eller whiteboard passer godt til denne lidt uformelle teknik. Man har nemmere ved at rette i pseudokode, og du kan vælge at bruge pseudokode, der ligner C#. Dvs. man kan sagtens bruge nøgleord som if-else, while, mm., og også { } for at angive kodeblokke, men du behøver ikke, og det må ikke være eksakt C#-kode; det skal jo ikke kompileres. Det kræver øvelse at finde det rette niveau.

Nu skal du i denne opgave ikke selv lave et SD, men når man selv laver et SD og ikke selv skal implemen­tere konkret kode, så er det vigtigt at overveje, om man giver programmøren tilstrækkelig information i SD’et til, at vedkommende reelt er i stand til at kode det, man ønsker, samtidig med at man ikke behøver at angive alting ned i mindste detaljer.

Du skal nu i de følgende øvelser lave pseudokode ud fra flere SD’er og et DCD.

Benyt **parprogrammering** til øvelserne.

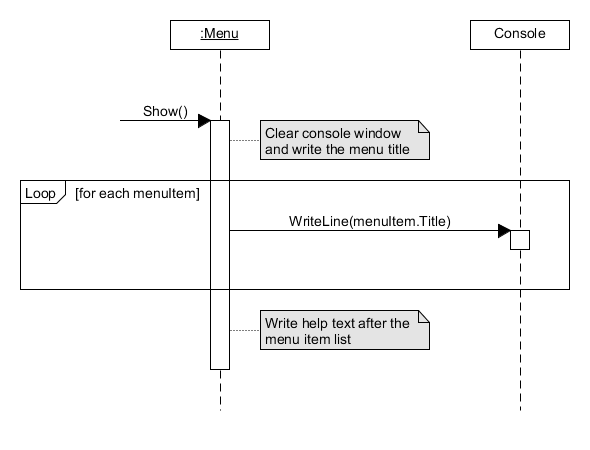
## Øvelse 3.1: Lav pseudokode for Main()-beskeden i Program-klassen

Med udgangspunkt i DCD’et og SD’et i øvelse 2, udfør følgende sammen med din makker:

* Lav pseudokode for Main()-beskeden i Program-klassen angivet i sekvensdiagrammet i øvelse 2.
  + Overvej hvilke trin der indgår i aktiveringssøjlen for Main().   
    Nedbryd, og beskriv de enkelte mere konkret i pseudokode.

## Øvelse 3.2: Lav pseudokode for Show()-beskeden i Menu-objektet

Du bruger samme DCD som før, men skal nu lave pseudokode for nedenstående SD.   
Læg mærke til brugen af kommentarer:

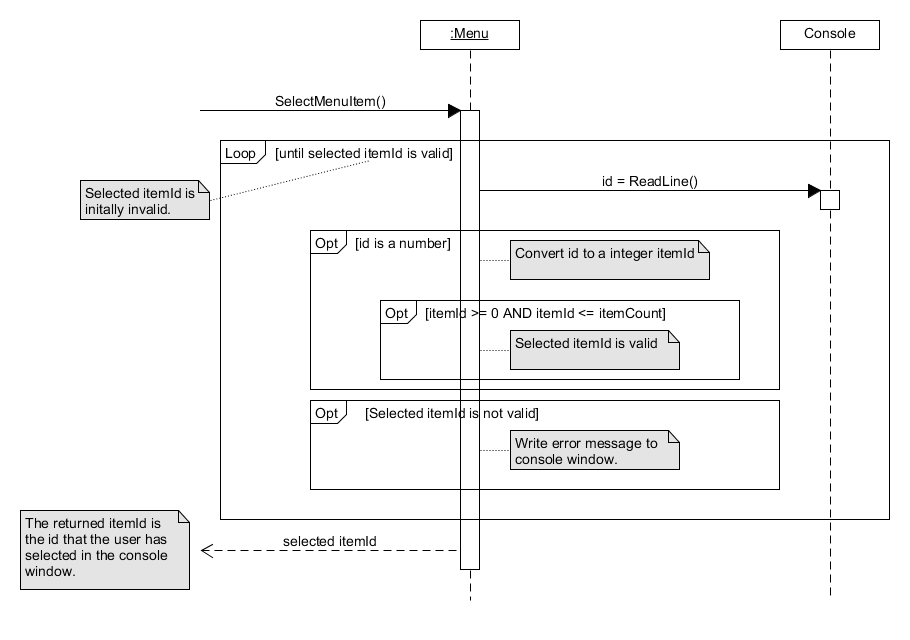


Udfør følgende sammen med din makker:

* Sammenlign SD’et med DCD’et.
* Lav pseudokode for Show()-beskeden i Menu-klassen angivet i sekvensdiagrammet foroven.

## Øvelse 3.3: Lav pseudokode for SelectMenuItem()-beskeden i Menu-objektet

Igen tager du udgangspunkt i samme DCD, men ser nu på følgende SD for SelectMenuItem()-beskeden:



Udfør følgende sammen med din makker:

* Lav pseudokode for SelectMenuItem()-beskeden i Menu-klassen angivet i sekvensdiagrammet foroven.

# Øvelse 4: Implementering af menu-system

I øvelse 2 i fredagsopgaven Ex07-Menu implementerede du din egen version af et menusystem, som brugte samme DCD, som i denne opgave. **Benyt fortsat parprogrammering til nedenstående øvelser.**

## Øvelse 4.1: Sammenlign pseudokode med din egen implementering

Udfør følgende:

* Sammenlign din pseudokode for Program.Main(), Menu.Show() og Menu.SelectMenuItem() og se, hvor din C#-kode fra Ex07-Menu adskiller sig fra pseudokoden, du lige har lavet.
* Hvad er anderledes eller det samme?
* Har pseudokoden opklaret nogle oprindelige misforståelser? Eller skabt nogen?  
  Notér dine fund.
* Overvej, om der er noget, der kunne gøres bedre eller anderledes.

## Øvelse 4.2: Tilret/færdiggør din implementering

Udfør følgende:

* Med udgangspunkt i dine overvejelser og sammenligninger, overvej om du kan forbedre og/eller færdiggøre din kode til menu-systemet fra opgave Ex07-Menu.
* Tilret/færdiggør din C#-kode til menu-systemet fra opgave Ex07-Menu, til du er tilfreds med resultatet.