# Conditionals (da: forgreninger)

|  |  |
| --- | --- |
| **Læringsmål** | Du kan:   * 1Pf1: anvende centrale metoder til at specificere og konstruere algoritmer [...] * 1Pf2: anvende centrale faciliteter i programmeringssproget til realisering af algoritmer […] |
| **Forventede  produkter** | * Implementering af programmeringsøvelserne |
| **Forventet læringsudbytte [SOLO]** | Designsporet:   * [Unistrukturel] Du kan identificere enkelte kendetegn ved **Computational Thinking** – dekomponering * [Unistrukturel] Du kan genkende enkelte kendetegn ved **strukturmodellering** – konceptuel klasse, klassenavn, attribut(navn)   Programmeringssporet:   * [Unistrukturel] Du kan genkende enkelte kendetegn i den **logiske struktur** af en applikation – kodeblok, sætning * [Unistrukturel] Du kan genkende enkelte kendetegn ved **datarepræsentation** – variabel-erklæring, virkefelt (en: scope) * [Unistrukturel] Du kan genkende enkelte kendetegn ved **datatyper** – int, bool, string * [Unistrukturel] Du kan genkende enkelte kendetegn ved **datahåndtering** af en variabel (tildeling, initialisering, konvertering, læsning) samt genkende og angive simple udtryk (en: expressions) med heltal (int), tekst (string) samt betingelser (en: conditions) via booleske udtryk (bool) * [Unistrukturel] Du kan huske enkelte kendetegn ved **programflow** – sætning, sekvens, forgrening (if-else) * [Unistrukturel] Du kan genkende enkelte kendetegn ved en **konsolapplikation** (Console.WriteLine, Console.Write og Console.ReadLine)   Færdighedssporet:   * [Unistrukturel] Du kan genkende enkelte kendetegn ved **kodning** * [Unistrukturel] Du kan genkende enkelte kendetegn ved **debugging** – ’Step Over’ * [Unistrukturel] Du kan genkende enkelte kendetegn ved **parprogrammering** |
| **Din forberedelse** | Som minimum til opgavens løsning, da læs tekst eller gennemse videoer markeret med **fed tekst** (eller start med dem). Det forventes, at du hjemme har gennemgået minimumsmaterialet, inden du møder op til undervisning, så du er parat til at løse opgaven sammen med dit team.  Nogle videoer er i forberedelsen angivet med et link (f.eks. til youtube).  Videoer uden et link kan i stedet findes i Planer i ItsLearning.  Se videoerne og læs teksterne i den rækkefølge, der er angivet forneden.  Læring:   * CL[[1]](#footnote-1): Genopfrisk Møde-på-midten og (B)Ordet-rundt * **SOLO[[2]](#footnote-2):** [**Understanding understanding**](https://www.youtube.com/watch?v=SfloUd3eO_M&feature=youtu.be&t=196) **(video: fra 3:15 til 6.18)**   Designsporet:   * [**Computational Thinking: Decomposition**](https://www.youtube.com/watch?v=yQVTijX437c) **(video: kun de første 6:23)** * [**Decomposition (BBC)**](https://www.bbc.co.uk/bitesize/guides/zqqfyrd/revision/1) **– side 1, 2 og 3 samt testen**   Programmeringssporet:   * **[YB] 1.3-1.4 (s. 9-13, 5 sider)** * Hvad er data? (video: 6:18) * Datarepræsentation:   + **Virkefelt for en variabel (video: 4:40)** * Logisk struktur:   + Kodeblok i kodeblok (video: 5:31) * Datatype:   + **Int (heltal) – literal værdi (video: 3:11)**   + **Bool – literal værdi (video: 2:54)** * Datahåndtering:   + Tildeling og udtryk (video: 7:53)     - **Booleske udtryk (video: 10:30)**     - [**Working with Operators and Control Flow in C#**](http://www.informit.com/articles/article.aspx?p=2002831&seqNum=5) **(forklarer datatypen bool)**   + **Initialisering (video: 3:23)**   + Konvertering:     - **String til heltal (video: 7:24)** * Program flow:   + [**The if Decision Statement**](https://channel9.msdn.com/Series/CSharp-Fundamentals-for-Absolute-Beginners/The-if-Decision-Statement) **(video: kun de første 15:15)**   Færdighedssporet:   * [Pair Programming](https://en.wikipedia.org/wiki/Pair_programming) * [**Agile in Practice: Pair Programming**](https://www.youtube.com/watch?v=ET3Q6zNK3Io) **(video: 3:11)** |

I denne opgave fortsætter du med anvendelse af data i din kode via variable, hvor du introduceres til to nye simple (og centrale) datatyper, nemlig int og bool, nu også med lidt mere fokus på, hvordan du repræsen­terer data, hvad en datatype er, og hvordan du håndterer data. Du introduceres også til en ny sætningstype (if-else) til at styre program­flowet, dvs. styre hvilke sætninger der udføres (og ikke udføres); denne sætningstype betegnes en forgrening (en: conditional).

Du skal også udføre en SOLO-øvelse og se nærmere på dekomponering i Computational Thinking (CT) samt introduceres til samarbejdsformen parprogrammering, som er en udbredt teknik til at minimere fejl, sikre øget forståelse og kvalitetskode. Tilsidst snuser du ganske kort til begrebet ’konceptuel klasse’. Selvom det lyder af meget, så husk at det er på unistrukturelt niveau (SOLO).

**Sørg for at have lavet øvelse 3, 4 og 6 i denne opgave, da du skal arbejde videre med dem i næste opgave Ex03-CSharpData.**

# Dagens ord: SOLO – og den personlige læringsudbytteplan

Som du nok har bemærket, så er der en sektion på forsiden, der hedder ”Forventet læringsudbytte [SOLO]”.

Som vi (underviserteamet) også forklarede på førstedagen, så er SOLO en taksonomi (inddeling), der er til hjælp for dig, så du kan italesætte dit læringsudbytte, og se hvor godt du er med i forhold til det forventede læringsudbytte for dagen.

Vi anbefaler, at du opretter et helt personligt dokument – enten lokalt på din maskine, eller på en cloud service (for eksempel Google Docs) – som du løbende holder ajour. Det er helt op til dig selv, hvor ofte du gør dette.

Vi anbefaler, at du benytter følgende skabelon, som er tænkt som et forslag til selvhjælp:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Emne** | **Uni-**  **strukturel** | **Multi-**  **strukturel** | **Relationelt** | **Udvidet**  **abstrakt** | **Begrundelse** | **Tiltag** |
| Løkker |  | X |  |  |  |  |
| Forgreninger |  |  |  |  |  |  |
| Datatyper |  |  |  |  |  |  |
| ... |  |  |  |  |  |  |

Du kan her holde styr på, hvilke emner der er gennemgået og hvor du selv vurderer dig til at være på SOLO-skalaen i forhold til emnet samt angive hvilke tiltag du planlægger for at øge dit SOLO-niveau.

# Øvelse 1: SOLO øvelse

Del teamet i to grupper.

## Øvelse 1.1: Placér beskrivelser på SOLO-niveau

Benyt **Ordet rundt** i hver gruppe, og kig på følgende læringsudbytte­beskrivelser en for en. Sørg for, at alle bliver hørt i gruppen. Vurdér og placér hver enkelt beskrivelse på det SOLO-niveau, du og din gruppe mener er rigtig:

* Du kan forklare Computational Thinking og begrunde din **forklaring**
* Du kan nævne enkelte kendetegn ved C# programmeringssproget
* Du forstår løkker og kan implementere løkker i flere programmeringssprog
* Du forstår løkker og kan vælge den optimale blandt flere løkketyper
* Du kan beskrive flere kendetegn ved SOLO

*Tidsramme: 15 minutter*

## Øvelse 1.2: Lav beskrivelser på hvert SOLO-niveau

Benyt **Ordet rundt** i hver gruppe, og find sammen beskrivelser for læringsudbytte for hver af de fire niveauer forneden:

* Unistrukturel
* Multistrukturel
* Relationel
* Udvidet abstrakt

Bemærk, at det ikke behøver at være inden for programmeringsområdet, men kan være helt generelt.

*Tidsramme: 15 minutter*

# Øvelse 2: Dekomponering i Computational Thinking (CT)

Dekomponering, eller nedbrydning, er en helt essentiel aktivitet for at håndtere en givet problemstilling. Oftest, hvis man har problemer med at formulere en algoritme, skyldes det, at man ikke har dekomponeret en problemstilling tilstrækkeligt.

## Øvelse 2.1: Terminologi

Del teamet i to grupper, og brug CL-strukturen **Møde på midten** til at reflektere over begrebet ”dekomponering”.

1. Reflektér over begrebet individuelt (1-2 min.)
2. Hvert gruppemedlem præsenterer dernæst resultatet af deres refleksion (1-2 min.)
   * Gruppen diskuterer det præsenterede (1-2 min.)
   * Bliv ved, indtil alle gruppemedlemmer er blevet hørt

*Tidsramme: 15 minutter*

## Øvelse 2.2: Dekomponér

Del teamet i to grupper, og vælg en passende CL-struktur.

Nedbryd følgende (overordnede) aktivitet:

* Rengør hele huset

Find så mange underaktiviteter du kan, dvs. så mange trin eller sekvenser af trin, det er muligt, til dækkende at beskrive den overordnede aktivitet.

*Tidsramme: 15 minutter*

## Øvelse 2.3: Saml trin i sekvenser

Ofte kan man samle flere trin i en sekvens og endda navngive sekvensen.

* Se i din nedbrydning foroven, om I kan gruppere relaterede trin i sekvenser og navngive dem.

*Tidsramme: 15 minutter*

# Øvelse 3: Programmering

I denne øvelse (og de resterende øvelser) benytter du dig af **parprogrammering** (forklaret i forberedelsen) sammen med en partner i teamet. *Hvis* I er et ulige antal til stede i dit team, skal I være tre i en gruppe.

Så find dig en partner, sæt dig ved siden af ham/hende ved en computer, og find ud af, hvem der skal skrive først, dvs. den der ’kører’ eller ’føreren’ (en: driver), og hvem skal observere eller navigere (en: navigator/observer). Husk at skifte rolle en gang imellem, så det ikke er den samme person, der programmerer under hele øvelsen.

## Øvelse 3.1: Terminologi

Del teamet i to grupper, og brug CL-strukturen **Møde på midten** til at reflektere over datatyperne ”int” og ”bool” samt begrebet ”udtryk (en: expression)”.

*Tidsramme: 10 minutter*

## Øvelse 3.2: Genopfrisk ’navn og alder’ med variable

Du skal sammen med din sidemakker kode en applikation, der skriver navn og alder ud til konsolvinduet. Mon du har set det før (genbrug er tilladt!) Denne gang skal du inddrage **både** string **og** int datatyper til de variable, hvor du midlertidigt gemmer og læser data.

Udfør følgende:

* Opret en ny konsolapplikation, igen med skabelonen: “Console App” (C#). Vælg selv et projektnavn.
* Erklær en variabel med navnet ’name’, der kan indeholde et navn; **vælg en datatype til tekst**
* Erklær en variabel med navnet ’age’, der kan indeholde en alder; **vælg en datatype til heltal (ikke string!)**
* Tildel literale værdier til de to variabler, f.eks. ”Jens” og 34.
* Udskriv til konsolvinduet (via reference til de to variable ’name’ og ’age’) formuleringen: ”Jens er 34 år gammel”, hvis navnet er Jens, og alderen er 34

*Tidsramme: 15 minutter*

## Øvelse 3.3: Brugerinput

I stedet for at tildele literale værdier for navn og alder til variabler i selve koden, da udvid jeres kode til at acceptere brugerinput, både tekst og tal.

Udfør følgende:

* Tilret koden til at indlæse navn og alder direkte fra konsolvinduet (genbrug fra tidligere opgave er tilladt)
  + Overvej, om der er noget fra forberedelsesmaterialet, der kan hjælpe dig med dette, specielt konvertering fra tekst til heltal.
* Inden du skriver navn og alder ud til konsolvinduet, træk da 10 fra alderen

*Tidsramme: 30 minutter*

# Øvelse 4: Forgreninger

Et centralt element i alle programmeringssprog er styring af programflowet, dvs. at styre rækkefølgen af sætninger eller kodeblokke, der bliver udført og (lige så vigtigt) ikke udført. Programmeringssproget C# (og stort set alle programmeringssprog) kan naturligvis bruges til at implementere algoritmer, hvis algoritmen er beskrevet tilstrækkelig detaljeret. Algoritmer har trin i CT, og i C# repræsenteres trin af sætninger. Du har allerede set eksempler på enkelte sætninger.

## Øvelse 4.1: Terminologi

Del teamet i to grupper, og brug CL-strukturen **Møde på midten** til at reflektere over begreberne ”betingelse”, ”booleske udtryk” og ”forgrening”.

1. Reflektér over begreberne individuelt (2 min.)
2. Hvert gruppemedlem præsenterer dernæst resultatet af deres refleksion (2 min.)
   * Gruppen diskuterer det præsenterede (2 min.)
   * Bliv ved, indtil alle gruppemedlemmer er blevet hørt

*Tidsramme: 10 minutter*

## Øvelse 4.2: Videre med navn og alder

I denne øvelse skal du arbejde med at styre valg gennem betingelser udtrykt via booleske udtryk, dvs. arbejde med at koble flere sætninger (CT: trin) sammen i kodeblokke (CT: sekvenser) og lade betingelser styre, hvilke sætninger der skal udføres. Konkret skal du anvende forgreninger via sætnings­typen if-else i C#.

Inden du går i gang, da overvej først, hvad programmet skal gøre (dekomponér). Med den viden du allerede har nu, overvej, hvilke data og datatyper der skal bruges, samt hvilken logik (betingelser og forgreninger) der er nødvendig til styring af programflowet.

Udfør da følgende:

* Opret et nyt konsolprojekt i Visual Studio

Skriv dernæst kode i Main()-metoden med følgende funktionalitet:

* Brugeren skal kunne indtaste sit navn og alder via konsolvinduet og gemme oplysningerne i henholdsvis en string og int variabel (husk konvertering). Genbrug kode fra tidligere øvelse.
* De indtastede oplysninger udskrives til konsolvinduet sammen med en angivelse, om man er et barn, teenager etc. ud fra den indtastede alder med følgende forslag til en aldersinddeling og -beskrivelse:
  + 0-12: ”et barn”
  + 13-19: ”en teenager”
  + 20-25: ”en studerende”
  + 26-67: ” i arbejde”
  + > 67: ”en pensionist”

Beskeden kunne have følgende fulde format:   
 ”Jens er 34 år gammel og er i arbejde” eller

”Anders er 72 år gammel og er en pensionist”

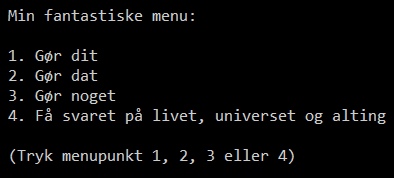
* Implementér programmet, og test det med forskellige navne og aldre

# Øvelse 5: Menu

Du har på nuværende tidspunkt byggestenene til at implementere en simpel menu-struktur og styre resultatet af menuvalget.

## Øvelse 5.1: Styring af menuvalg

Først benyt de funktioner, som du allerede kender nu til at skrive følgende menu ud på konsolvinduet:



Dernæst få styr på logikken, dvs. styring af menuvalget og udskrivning af resultatet:

* Afhængig af, hvad brugeren har trykket, da skriv følgende ud på konsolvinduet:
  + Hvis ’1’, skriv ”Punkt 1 er valgt: Gør dit”
  + Hvis ’2’, skriv ”Punkt 2 er valgt: Gør dat”
  + Hvis ’3’, skriv ”Punkt 3 er valgt: Gør noget”
  + Hvis ’4’, skriv ”Punkt 4 er valgt: 42”
  + Ellers skriv ”Forkert valg”
* Implementér og test programmet

## Øvelse 5.2: Tilpasning

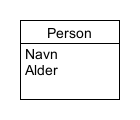
Inspicér din kode, og overvej om, der noget der kan gøres nemmere, smartere eller mere kompakt.

Skriv følgende to linjer ind sidst i din kode, og tilret den foregående kode, så det passer med nedenstående:



# Øvelse 6: Inspektion og kendetegn

Med udgangspunkt i dagens øvelse, kan du på nuværende tidspunkt beskrive en person som værende *noget*, der har et navn og en alder og som findes i den virkelige verden. Denne information kan beskrives i en *konceptuel klasse*[[3]](#footnote-3). Man kan visualisere en konceptuel klasse for en person som følgende:



* Lav en inspektion af ovenstående konceptuelle klasse (Person)
* Overvej, om der er noget andet, der kendetegner en person – lav en liste
  + Benyt gerne jeres whiteboard til denne øvelse

Vi vil se på, hvordan vi kan repræsentere denne information i C# i næste uge.

# Øvelse 7: Review spørgsmål

* Hvad står SOLO for?
* Hvorfor er SOLO vigtigt for dig?
* Hvad indebærer dekomponering?
* Hvad er en forgrening (en: conditional)?
* Hvad kan man benytte datatypen bool til?
* Hvad er et virkefelt (en: scope)?
* Hvilke roller er der i parprogrammering, og hvilke ansvar har de?
* Hvordan konverterer man en string til en int?
* Hvilke typer af operators findes der?
* Hvad er en konceptuel klasse?

1. CL = Cooperative Learning (se i ItsLearning i folderen ’1. Studieår’ --> ’Cooperative Learning Strukturer’) [↑](#footnote-ref-1)
2. SOLO = Structure of the Observed Learning Outcome [↑](#footnote-ref-2)
3. En klasse der beskriver noget fra den virkelige verden, der kan identificeres med hjælp af navneord (ord du kan sætte *en* og *et* foran. For eksempel en person, et hus og en kat). [↑](#footnote-ref-3)