# 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 de to programmeringsøvelser |
| **Forventet læringsudbytte [SOLO]** | Designsporet:   * [Unistrukturel] Du kan identificere enkelte kendetegn ved **Computational Thinking** – dekomponering (en: decomposition)   Programmeringssporet:   * [Unistrukturel] Du kan genkende en kodeblok i en C# **programstruktur** * [Unistrukturel] Du kan genkende enkelte kendetegn ved **datahåndtering** af en variabel (erklære, initialisere, tildele, læse, konvertere), aflæse dets virkefelt (en: scope) samt genkende og angive simple udtryk (en: expressions) med heltal (int), tekst (string) samt betingelser (en: conditions) via booleske udtryk (bool) * [Unistrukturel] Du kan genkende enkelte kendetegn ved simple C# **datatyper** (bool, int & string) * [Unistrukturel] Du kan huske enkelte kendetegn ved **program flow** – sekvens, if-else   Færdighedssporet:   * [Unistrukturel] Du kan genkende enkelte kendetegn ved **parprogrammering** |
| **Din forberedelse** | Læring:   * CL[[1]](#footnote-1): Genopfrisk Møde-på-midten og 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:   * [The if Decision Statement](https://channel9.msdn.com/Series/CSharp-Fundamentals-for-Absolute-Beginners/The-if-Decision-Statement) (video: 23:42) * [How to Convert a String to an Int in C#](https://www.youtube.com/watch?v=sBYfIpECQec) (video: 6:44, spring gerne ‘Solution 3’ over, vi kommer til try-catch sætningen senere) * [Working with Operators and Control Flow in C#](http://www.informit.com/articles/article.aspx?p=2002831&seqNum=5) (forklarer datatypen bool)   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 introduceres du til at styre program flowet (dvs. hvilke C#-sætninger der udføres) ved hjælp af forgreninger samt til begrebet ’konceptuel klasse’. Du skal også udføre en SOLO-øvelse og se nærmere på dekomponering i Computational Thinking (CT). **Sørg for at have lavet øvelse 3 og 4 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 måle, hvor godt du er med i forhold til det forventede læringsudbytte for dagen.

Vi (underviserteamet) anbefaler, at du opretter et 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 (underviserteamet) anbefaler, at du benytter følgende skabelon:

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

# 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 – relationel**
* Du kan nævne enkelte kendetegn ved C# programmeringssproget **– Multi struktural**
* Du forstår løkker og kan implementere løkker i flere programmeringssprog **– Udvidet abstrakt**
* Du forstår løkker og kan vælge den optimale blandt flere løkketyper **– relatioel**
* Du kan beskrive flere kendetegn ved SOLO **– Multi struktural**

*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 **– man lære at identificere, minimale datatyper såsom et string, og har løse ender indenfor området.**
* Multistrukturel **– Man lære at identificere og beskrive, flere datatyper såsom string & int, har forståelse for brugen af dem, dog stadig usamhængende.**
* Relationel **– Man kan se, identificere og forstå sammenhænget mellem flere datatyper. Her ka man bruge string og in i programmering og forstå hvorfor man benytter dem – og dermed har sammenhænget med.**
* Udvidet abstrakt **– at man kan bruge alt sin viden om string og int, og viderebringe det til fx andre programmeringssprog.**

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 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 (en: driver), og hvem skal observere (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 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: 20 minutter*

Et centralt element i alle programmeringssprog er styring af program flowet, dvs. at styre rækkefølgen af sætninger eller kodeblokke, der bliver udført og (lige så vigtigt) ikke udført. I de følgende underøvelser skal du arbejde med at styre valg gennem betingelser udtrykt via booleske udtryk.

Programmeringssproget C# 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. I underøvelserne forneden skal du arbejde med at koble flere sætninger (CT: trin) sammen i kodeblokke (CT: sekvenser), at styre gennem betingelser (beskrevet med booleske udtryk), og hvilke sætninger der skal udføres via forgreninger (C#: if-else).

Inden du går i gang med hver underøvelse, 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 program flowet.

## Øvelse 3.2: Videre med navn og alder

Udfør følgende:

* Opret et nyt konsolprojekt i Visual Studio

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

* Brugeren skal kunne indtaste sit navn og alder via konsolvinduet og gemme oplysningerne i henholdsvis en string og int variable (husk konvertering).
* 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”

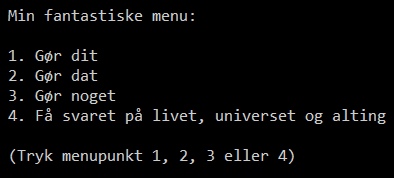
* Implementér programmet, og test det med forskellige aldre

## Øvelse 3.3: Menu

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

### Øvelse 3.3.1: 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 3.3.2: Tilpasning

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

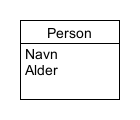
Skriv følgende to linjer ind sidst i din kode, og tilret den foregående kode, så det passer med nedenstående:



# Øvelse 4: 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 5: Review spørgsmål

* Hvad står SOLO for?
* **Hvorfor er SOLO vigtigt for dig?**

Structure of the observerved learning outcome

For at få en struktureret faglig forståelse for din læring

* **Hvad indebærer dekomponering?**

Nedbryder emner ind til mindre og bedre god bider så det er lettere at tage hånd om.

* **Hvad er en condition/forgrening?**

Det er hvis dette kriterie er mødt – gør dette, ellers ->

* **Hvad kan man benytte datatypen bool til?**

Benyttes for at finde sandt eller falsk.

* **Hvad er et virkefelt/scope?**

Inde i kodeblokken { }, variablen virker kun inde i virkefeltet (scope)

* **Hvilke roller er der i parprogrammering, og hvilket ansvar har de?**

Driver – og navigatør -> en skriver, den anden dirigere, så der er 2 til en opgave, minimere fejl.

* **Hvordan konverterer man en string til en int?**

STRING = Convert.ToInt32(Console.ReadLine());

Eller

int.TryParse(STRING, out int MIDLER\_TIDIG\_INT); ->

int DINNYEINT = MIDLER\_TIDIG\_INT;

* **Hvilke typer af operators findes der?**
* **Hvad er en konceptuel klasse?**

En klasse der beskriver noget fra den virkelig verden såsom en person ( udseende, religion osv)

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)