Methods (da: Metode)

Læringsmål	Du kan: • **1Pf1:** anvende centrale metoder til at specificere og konstruere algoritmer [...] • **1Pf2:** anvende centrale faciliteter i programmeringssproget til realisering af algoritmer […] • **1Pf3**: anvende et i professionen udbredt, integreret udviklingsværktøj, herunder versionsstyringssystem […] til at designe og konstruere praksisnære applikationer […] • **1Pk3**: i en struktureret sammenhæng tilegne sig ny viden, færdigheder og kompetencer inden for programmeringssprog, udviklingsværktøjer, programmeringsteknikker og programdesign
Forventede produkter	• Et C# konsolprogram der indeholder basale lommeregner funktioner (plus, minus, gange og divider)

Din forberedelse Programmeringssporet:

- Access Modifiers (C# Programming Guide)
  - Med fokus på public og private
- Genlæs parprogrammering
- Om objekt orientering (W3CSchools)
  - Class members
  - Class constructors
  - Class Access Modifiers (Fokus på public og private)
  - C# Indkapsling (Encapsulation)
- Grundlæggende klasse-diagram (DCD)

Færdighedssporet:

- Skimlæs navigating through code with the debugger

Beskrivelse af dagens opgave

Du blive introduceret til C# klasser, felter og metoder.

Sørg for at lave øvelserne færdige (Lommeregner) i denne opgave færdig, da du skal arbejde videre med denne øvelse om en uge-

Øvelse 1: Terminologi

Inden du skal i gang med dagens konkrete programmeringsopgaver, skal du have testet din forforståelse af dagens emne (samt fastholdt et par af de tidligere nævnte emner).

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

1. *C# metode*
    - *Med fokus på: access modifier, returtype, metodenavn, parameter*
2. *C# klasse, attributter, objekt (en: object), felt, konstruktør (en: constructor)*
3. *Object-Oriented Programming (OOP)*
    - *Med fokus på: indkapsling (en: encapsulation)*
4. *while, do-while, for-loop*

Fremgangsmåde:

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

Tidsramme: 15 minutter

Øvelse 1: Lommeregner

Du skal i denne øvelse inspicere en simpel use case og tilhørende designklassediagram (DCD). Herefter skal du oprette et lommeregnerprojekt med en lommeregner-klasse. **Undervejs skal du benytte dig af debugging**.

Øvelse 1.1: Inspicér use case og designklassediagram (DCD)

Du og dit team skal benytte jer af en passende CL-struktur til at inspicere følgende use case og designklassediagram.

Use case: Udregn regnestykke

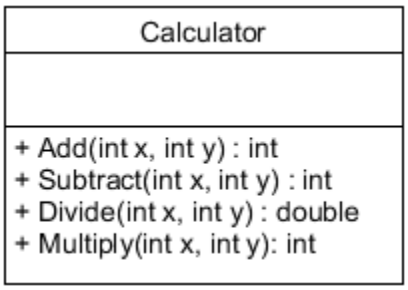
Primæraktør: Beregneren

Main Success Scenario:

1. Beregneren modtager et regnestykke
2. Beregneren starter ny beregning
3. Beregneren indtaster tal
4. Beregneren vælger regneoperator (+, -, *, /)
   Beregneren gentager trin 3 og 4, indtil hele regnestykket er indtastet
5. Systemet beregner og viser resultatet

Designklassediagram: Calculator

Med fokus på en enkelt software designklasse:



Øvelse 1.2: Lommeregnerprojekt

Benyt parprogrammering.

I denne øvelse skal du oprette og benytte en simpel lommeregner. Hertil skal du oprette et lommeregnerprojekt, din første C# klasse, dine første C# metoder, samt benytte dig af en ny debugging kommando (Step Into).

Øvelse 1.2.1: Opret lommeregnerprojekt og lommeregner klasse

1. Opret en ny konsolapplikation
    a. Navngiv det ”Lommeregner”
2. Tilføj en ny C# klasse til din konsolapplikation
    a. Højre-klik på dit projekt ”Lommeregner” i din Solution Explorer, vælg Add, vælg Class, giv klassen navnet ”Calculator”.

Øvelse 1.2.2: Opret lommeregner metoder

1. Du skal nu i Calculator-klassen oprette metoderne ”Add”, ”Substract”, ”Divide” og ”Multiply” (se også ovenstående DCD fra øvelse 3.1)
    - Add, skal lægge to tal sammen og returnere dette som et heltal
    - Subtract, skal trække to tal fra hinanden og returnere dette som et heltal
    - Divide, skal dividere to tal og returnere dette som et decimaltal
    - Multiply, skal gange to tal med hinanden og returnere dette som et heltal

Øvelse 1.2.3: Benyt lommeregner

1. Du skal nu skifte over til din Main-metode i din Program-klasse (Program.cs). Her skal du lave en ny instans af din Calculator-klasse og benytte en af objektets metoder og printe resultatet ud på konsolvinduet

Øvelse 1.2.4: Benyt debugging

Du har allerede fået introduceret dette vigtigt redskab i tidligere, men denne gang skal du benytte dig af ’Step Into’, frem for ’Step Over’.

1. Benyt debugging ’Step Into’ (F11)

Øvelse 2: Mere menu

Du skal overveje ud fra den resterende tid, om du og dit team har tid til denne øvelse.

Øvelse 2.1: Opret menu med hjælp af en do-while løkke

1. Hvis du stadig har tid, så lav en menu i konsolapplikationen, benyt do-while løkke.
    - Det skal være muligt at benytte alle metoderne på skift.

Øvelse 2.2: Indtaste tal og modtage svar

Du skal gøre det muligt for en bruger at vælge en af metoderne (Add, Multiply, Divide eller Substract), samt indtaste to tal og modtage svar tilbage på skærmen.

Krav til implementeringen:

- Du skal benytte følgende datatyper:
  - double
  - int
  - String
- Benyt Console.ReadLine();
- Benyt 'if-else' eller en 'switch'
- Du skal sikre dig imod forkert input

Øvelse 3: Review-spørgsmål

- Skal man definere en datatype ved metodedefinitionen (formelle)?
- Kan man have mere end en inputparameter?
- Hvad tænker man på ved returtypen?
  - Hvordan sikrer man at metoden returnerer noget?
  - Hvad hvis man ikke vil returnere noget?
- Skal man definere en datatype ved metodeskaldet (aktuelle)?
- Kan man kalde ens metode mere end en gang?
- Er der forskel på "git add" og "git commit"?
- Hvad er forskellen mellem "git pull" og "git push"?
- Hvad gør 'Step Into'?
- Hvad gør 'Step Over'?