# Unit test and array

|  |  |
| --- | --- |
| **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 * **1Pf6**: anvende moderne teknikker og værktøjer til afvikling af test […] |
| **Forventede  produkter** | * Et UnitTestet C# konsolprogram * Implementering af array-programmeringsøvelser |
| **Forventet læringsudbytte [SOLO]** | Programmeringssporet:   * [Unistrukturel] Du kan genkende enkelte kendetegn ved **datatyper** - int, string, bool, double, char, array * [Unistrukturel] Du kan huske enkelte kendetegn ved **programflow** – sekvens, forgrening (if-else, switch), løkke (while, do-while, for), metodekald og -retur, exception handling (try-catch-finally), jump statements (break, continue)   Færdighedssporet:   * [Unistrukturel] Du kan genkende enkelte kendetegn ved **testing** – strukturen af en unit test i Visual Studio IDE (Test class attributes, test method, assertions) |
| **Din forberedelse** | Læring:   * [SOLO-taksonomien – et stærkt afsæt for et fælles sprog om læring](https://www.kompetencehusetheckmann.dk/solo-taksonomien-staerkt-afsaet-faelles-sprog-laering)<https://www.kompetencehusetheckmann.dk/solo-taksonomien-staerkt-afsaet-faelles-sprog-laering/>   Programmeringssporet:   * [The char type](https://csharp.net-tutorials.com/data-types/the-char-type/) * [A basic introduction of Unit Test for beginners](https://www.c-sharpcorner.com/article/a-basic-introduction-of-unit-test-for-beginners/)   + Du skal ikke udføre eksemplet, men blot forholde dig til det   + **Bemærk:** Vi benytter os af .Net 6 og **ikke** .Net Framework   + Skip den sidste del omkring Test Driven Development [TDD] * [Inspect variables in the Autos and Locals windows](https://docs.microsoft.com/en-us/visualstudio/debugger/autos-and-locals-windows?view=vs-2019) * [YB] ’Breaking Out of Loops’ og ’Going back to the top of a loop’ (s. 42-43)   + Også kendt som jump statements * [YB] 3.2.2 (s. 59-60) Nested Blocks   + Med fokus på ’For loop local variables’ * [YB] 3.3 – 3.3.3 (s. 61-64) Arrays * [YB] 3.4 – 3.4.4 (s. 66-69) Exceptions |

Du har lært, hvordan man kan gemme enkeltstående informationer, såsom navn, alder og telefonnummer. Men hvad, hvis du har tænkt dig at gemme og fremvise en liste af disse enkeltstående informationer, hvordan vil du gøre det? Det skal vi se nærmere på i denne opgave.

**Sørg for at gøre øvelse 3 (Lommeregner) i opgaven Ex05-Methods færdig som forberedelse, da du arbejder videre med denne øvelse i denne opgave.**

# Dagens citat:

[Donald Knuth](https://en.wikipedia.org/wiki/Donald_Knuth): “*Computers are good at following instruction, but not at reading our mind*”.

# Øvelse 1: Terminologi

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

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. *SOLO niveauerne (unistrukturelt, multistrukturelt, relationelt, udvidet abstract)*
2. *Array (element, index), char (numeric representation)*
3. *Exception (try-catch), jump statement*
4. *Unit Test (TestClass attributes, TestMethod, assertion)*

**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 (2 minutter)
5. Team: Præsentér på skift det, som I er blevet enige om, til hele teamet (uret rundt) (2 minutter per gruppe)
   1. De øvrige omkring bordet tager noter og stiller spørgsmål

*Tidsramme: 20 minutter*

# Øvelse 2: Lommeregnerprojekt genvisit

I denne øvelse skal du teste dit tidligere udarbejdede lommeregnerprojekt fra Ex05-Methods, hertil skal du tilføje et MSTest Test Project, hvilket er et cross platform Unit Test framework. Hvis du **ikke** nåede at blive færdig med at udarbejde lommeregneren og de fire metoder (add, substract, divide og multiply), gå da tilbage til forrige opgave, og lav øvelse *Øvelse 3* inden du går videre – og husk at bede om hjælp, hvis du eller dit team har behov for dette.

## Øvelse 2.1: Sikring af at din lommeregner klasse er Public

Du skal åbne dit lommeregnerprojekt, som du udarbejdede sidste gang (Ex05-Methods).

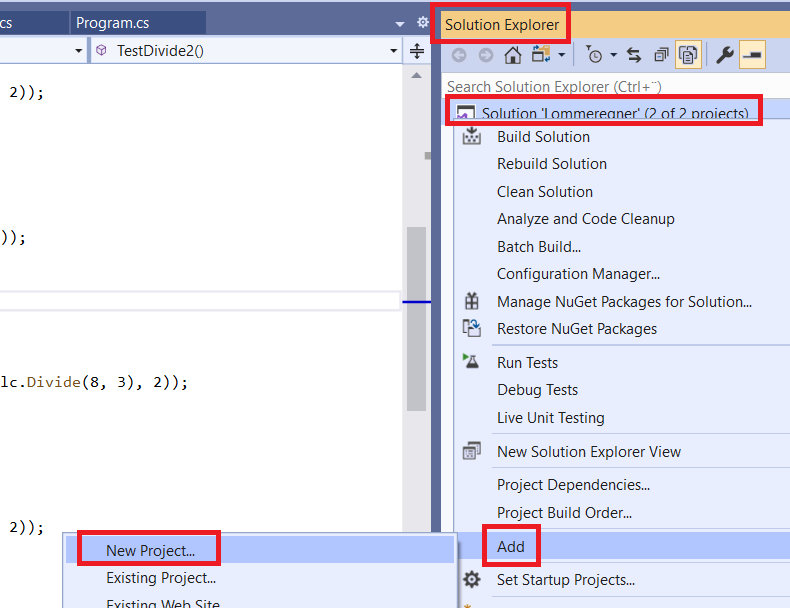
Du skal herefter modificere din Calculator.cs-fil ved at tilføje **public** foran ”class Calculator”, som på billedet nedenunder:

Dette gør, at dit Calculator-klasse kan benyttes andre steder i dit projekt. I dette tilfælde er vi interesseret i, at vores MSTest Test Projekt kan se vores klasse.

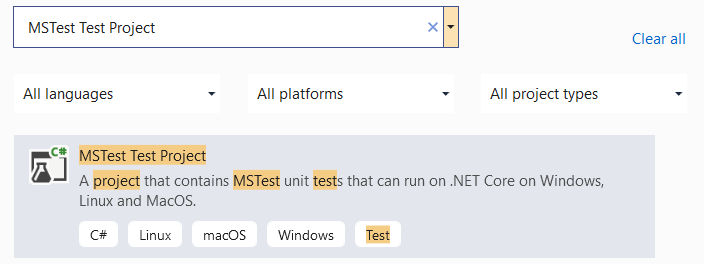
## Øvelse 2.2: Tilføjelse af Unit Test projekt

Du skal du tilføje et nyt projekt til din solution, et MSTest Test Project.

Først skal du højreklikke på din *Solution* (**ikke** på dit lommeregnerprojekt) i ”Solution explorer”, vælg ”Add”og endeligt ”New project…”.



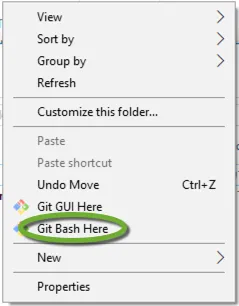
Søg herefter efter ”MSTest Test Project”, som på billedet nedenunder:



Navngiv testprojektet: LommeregnerTest

## Øvelse 2.3: Klon materialefolder

Der er en mappe med supplerende materiale tilknyttet programmerings-faget. Mappen befinder sig inde i et Git repository, og du vil fremover modtage opgavespecifikke filer herinde. Det vil ofte være Unit Tests, hvilket det også er i dag. Du skal i denne øvelse lave en klon af dette Git repository.

Åbn Git Bash i den mappe, hvor du vil placere mappen med det supplerende materiale. Højreklik og vælg ”Git Bash Here”, som på følgende billede:

Benyt følgende kommando i Git Bash:

* git clone <https://gitlab.com/GonzoDark/dmu-2022-2023-exercise-materials.git>

Hvis alt er gået som det skulle, burde du nu gerne have en mappe med navnet ”dmu-2022-2023-exercise-materials”. Hvis du åbner ”dmu-2022-2023-exercise-materials”, burde du gerne have filerne ”*Ex06-UnitTestAndArray.UnitTests.docx*” og ”*Ex06-UnitTestAndArray.UnitTests.Bonus”*. Du skal bruge disse filer i følgende øvelse.

## Øvelse 2.4: Kopiering af Unit Tests fra materialefolderen

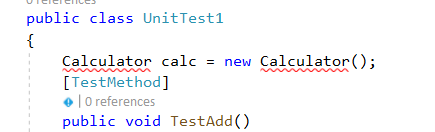
Nu er du klar til at benytte den ene af de to filer, som du hentede i forrige øvelse.

Åbn “*Ex06-UnitTestAndArray.UnitTests.docx*”. Du skal kopiere indholdet fra filen ind i din UnitTest1.cs-fil i dit Unit Test projekt, vær opmærksom på scope, og *du skal ikke overskrive hele filen*, kun din TestClass med tilhørende TestMethods.

### Øvelse 2.4.1: Inspektion af program-tilstand efter kopiering

Bemærk, at du vil se følgende fejl i din Error List:

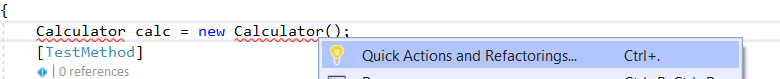
* *The type or namespace name ’Calculator’ could not be found (are you missing a using directive or an assembly reference?)*

Og du vil se en rød streg under din ”Calculator”-klasse.

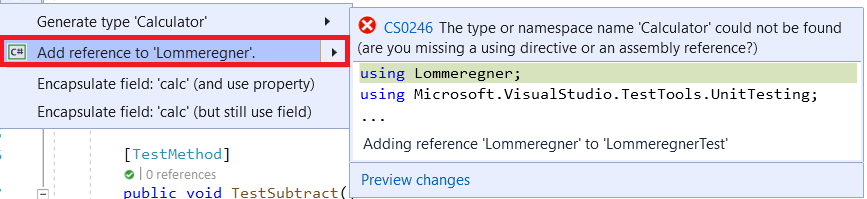
Dette er, fordi du mangler en reference til dit lommeregnerprojekt, det vil vi se på i følgende del-øvelse (øvelse 2.5).

## Øvelse 2.5: Oprettelse af reference til lommeregnerprojekt

Forudsat, at du har ændret din Calculator-klasse til at være public, og at den rent faktisk hedder Calculator (både klassen og filen), kan du nu tilføje en reference ved at højreklikke på ”Calculator” og vælge ”Quick Actions and Refactorings…”.

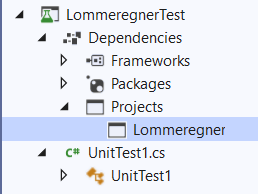


Og derefter ”Add reference to ’Lommeregner’”:



Brug også ”Quick Actions and Refactorings” på Math.Calc i unittesten ”TestDivide2” (Hvis der er behov for dette).

### Øvelse 2.5.1: Inspektion af Unit Test projekt

Nu burde du også kunne se ”Lommeregner” under Projects i dit Unit Test projekt, som på billedet nedenunder (fold ”Dependencies” ud, samt ”Projects”):

|  |
| --- |
| Hvis du **ikke** kan se referencen (og du har foldet Projects ud), så højreklik på ”Dependencies” i dit Unit Test Projekt, og vælg ”Add Project Reference…”, som ses her:  Og vælg dit lommeregner-projekt under ”Projects --> Solution”, som ses her: |

## Øvelse 2.6: Benyt Test Exploreren

Nu er du endelig klar til at benytte Test Exploreren.

Fremgangsmåde til at køre Unit Test projektet:

* Tryk på ”Test” i hovedmenuen --> ”Windows” --> ”Test Explorer” (grøn knap til venstre, ”Run All”)

Bemærk, at den ene af dine tests burde være rød (TestDivide2), resten burde være grønne.

|  |
| --- |
| *Hmmm.. TestDivide2 er* ***ikke*** *grøn:* Overvej, hvordan du kan få din Unit Test til at lykkedes. Tilret din løsning, og forsøg igen, indtil alle tests er grønne (*måske kan informationssøgning/en google søgning efter typecasting hjælpe med TestDivide2?*). |

# Øvelse 3: Færdighedsøvelser

Det er tanken med disse underøvelser, at du bliver lidt klogere på forskellene mellem at arbejde med og uden arrays, samt snuse lidt til exception handling og parsing. Tidligere introducerede emner (fra tidligere programmeringsopgaver) vil også komme i spil.

Du skal i underøvelserne gå sammen med din sidemand og benytte parprogrammering (Ja, det vil sige, at én er driver og én er navigatør/observer[[1]](#endnote-1)).

## Øvelse 3.1: Alderen på alle i gruppen, samt aldersgennemsnit (uden array)

Din opgave er at printe alderen af alle ved bordet samt printe gennemsnitsalderen ud til skærmen. Du må **ikke** benytte dig af et array til denne opgave.

**Fremgangsmåde:**

* Erklær en række af variabler til at indeholde alderen for hver person ved dit bord
* Tildel variablerne (aldrene) retvisende værdier
* Print aldrene ud til skærmen, en alder per linje
* Erklær en variabel til at kunne indeholde gennemsnitsalderen af alle ved dit bord
* Beregn og print gennemsnitsalderen ud til skærmen

## Øvelse 3.2: Alderen på alle i gruppen, samt aldersgennemsnit (med array)

Denne øvelse foregår på samme måde som sidste øvelse (3.1). Du skal modsat sidste opgave benytte dig af et array til denne opgave samt en passende løkke (en: loop).

Bemærk, at du i det følgende **ikke** må anvende de indbyggede funktioner Sum() og Average() i array-typen til at løse opgaven. Dvs. du skal selv implementere beregningen af aldersgennemsnittet.

**Fremgangsmåde:**

* Erklær et array med en passende størrelse, så det kan indeholde alle aldre ved bordet (og af en passende datatype)
* Tilføj alderen af alle ved bordet til arrayet
* Erklær en variabel til at kunne indeholde gennemsnitsalderen af alle ved dit bord
* Benyt en løkke (en: loop) til at printe alle aldrene fra arrayet ud til skærmen
* Beregn og print gennemsnitsalderen ud til skærmen

Overvej herefter: Er der en forskel i længden af koden i denne øvelse (3.2) kontra den tidligere (3.1)?

### Øvelse 3.2.1: Erklær og tildel

Inspicér følgende artikel med henblik på det første kodeeksempel: [Arrays](https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/arrays/)

Lav herefter om på dit program, så du både erklærer og tildeler dit arrays elementer i samme omgang.

## Øvelse 3.3: Findes der en med alderen

Bemærk, at du i det følgende IKKE må anvende den indbyggede funktion Contains() i array-typen til at løse opgaven. Dvs. du skal selv implementere søgningen.

Din opgave er at udvide programmet fra ovenstående øvelse (3.2.1). Du skal tilføje en søgefunktion, hvor det er muligt at søge efter, om der findes en bestemt alder i arrayet. Under din implementering skal du benytte dig af ”break”.

Når du er færdig med implementeringen, skal du overveje, hvorfor det er smart at benytte et break – diskuter dette med din sidemand.

## Øvelse 3.4: Bestemt antal personer

Udvid løsningen fra øvelse 3.3. Det skal være muligt for en bruger af dit program at definere antallet af personer, der er i en givet gruppe – for dernæst at tilføje deres aldre. Du skal med andre ord erklære en variabel til at indeholde brugerinput omkring antallet af personer i en gruppe, for derefter at sætte størrelsen af dit array til at være lig med denne variabel.

### Øvelse 3.4.1: try-catch

Du skal benytte try-catch til at håndtere, hvis en bruger skriver bogstaver, frem for heltal.

### Øvelse 3.4.2: int.TryParse

Frem for at benytte dig af try-catch, som i øvelse 3.4.1, skal du denne gang læse omkring [int.TryParse](https://docs.microsoft.com/en-us/dotnet/api/system.int32.tryparse?view=net-5.0) og benytte det i din løsning.

# Bonusøvelse 4: Sokker og lys

Udfør kun følgende del-øvelser, hvis du og dit team har tid.

## Øvelse 4.1: Par sokker

Benyt parprogrammering til at løse opgaven.

Du skal forestille dig, at der foran dig er en stor stak sokker. Hver sok har en unik farve, som du skal repræsentere med et heltal. Din opgave er at lave en metode, der kan returnere, hvor mange par der er i bunken med samme farve. Du skal benytte dig af et array under din implementering.

|  |
| --- |
| Hvis du **ikke** allerede har hentet Unit Testene, så benyt ”git pull” i din Git mappe, se filen *Ex06-UnitTestAndArray.UnitTests.Bonus*. Testene er navngivet ”SockMerchantX”. Få alle tre tests til at blive grønne. |

## Øvelse 4.2: Puste lys ud

Benyt parprogrammering til at løse opgaven.

Du skal forestille dig, at du er med til en fødselsdagsfest, hvor fødselaren kun er i stand til at puste de højeste lys ud på kagen. De lys, der er på kagen, er repræsenteret af et heltalsarray, hvor 1 er det laveste lys, og jo højere tallet er, jo højere er lyset. Din opgave er at lave en metode, der kan returnere antallet af lys, som fødselaren har pustet ud.

|  |
| --- |
| Hvis du **ikke** allerede har hentet Unit Testene, så benyt ”git pull” i din Git mappe, se filen *Ex06-UnitTestAndArray.UnitTests.Bonus*. Testene er navngivet ”CandleBlowerX”. Få alle tre tests til at blive grønne. |

1. <https://en.wikipedia.org/wiki/Pair_programming> [↑](#endnote-ref-1)