# Orange Tree

|  |  |
| --- | --- |
| **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 |
| **Forventet læringsudbytte [SOLO]** | Programmeringssporet:   * [Unistrukturel] Du kan genkende enkelte kendetegn i **programstrukturen** – sætning, kodeblok, metode (navngivet sekvens), klasse, namespace, løsning (en: solution), projekt * [Unistrukturel] Du kan genkende enkelte kendetegn ved **C#** **klasse (objekt)** – C# klasse, felt (en: field), metode, access modifiers (public, private), access keywords (this) * [Unistrukturel] Du kan genkende enkelte kendetegn ved **navngivningskonventioner** (en: naming conventions)   Færdighedssporet:   * [Unistrukturel] Du kan genkende enkelte kendetegn ved **testing** – strukturen af en unit test i Visual Studio IDE (Unit testing attributes (Test class, test method, Test initialize), Unit test assertions) |
| **Din forberedelse** | Programmeringssporet:   * [this (C# Reference)](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/this) * [Test Initialize and Test Setup](https://www.c-sharpcorner.com/UploadFile/dacca2/test-initialize-and-test-setup/) * [C# - Namespaces](https://www.tutorialspoint.com/csharp/csharp_namespaces.htm) * [C# Namespaces](https://www.youtube.com/watch?v=AK40rIzwDbU) (video: 5:51) * [C# Coding Conventions](https://docs.microsoft.com/en-us/dotnet/csharp/fundamentals/coding-style/coding-conventions#naming-conventions) |

Du skal i denne opgave lave en lille simulator for et appelsintræs livsforløb. Øvelserne i opgaven illustrerer processen med at indkapsle ens interne datastruktur, definere en kontrolleret tilgang til sin datastruktur, implementere træets adfærd samt teste implementeringen med en unit-test. Så du genbesøger tidligere emner med et nyt eksempel og ny vinkel. Forberedelsen introducerer navngivningskonventioner (en: naming conventions) og namespaces, som også illustreres i opgaven.

# Øvelse 1: Terminologi

Del teamet op i mindre grupper (af 2 oersiber), og brug **Ordet rundt** til at reflektere over begreberne ”navnekonventioner (en: naming conventions)”,”test initialize”, ”this keyword” og ”namespaces”. Sørg for, at alle får mulighed for at tale.

*Tidsramme: 20 minutter*

# Øvelse 2: OrangeTree-simulation

De første øvelser forneden omfatter oprettelse af konsolprojekt, testprojekt og implementering af den centrale C#-klasse, OrangeTree, til simulering af et appelsintræs livsforløb. Den sidste øvelse er selve gennem­førelse af en unit-test for at tjekke, om din implemente­ring er korrekt. Når du kører testen i Test Explorer, retter du i din implementering, indtil alle tests kører uden problemer (dvs. er grønne). Det forklares nærmere, når du kommer til øvelsen.

## Øvelse 2.1: Forberedelse af konsolprojekt og testprojekt

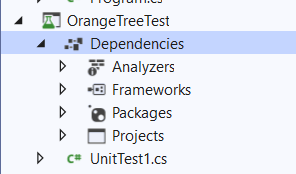
Udfør følgende:

* Opret et nyt konsolprojekt med navnet ”OrangeTreeSim”
* Opret i samme solution et nyt unit-test projekt (projektskabelon: ”*MSTest Test Project*”) med navnet ”OrangeTreeTest”.

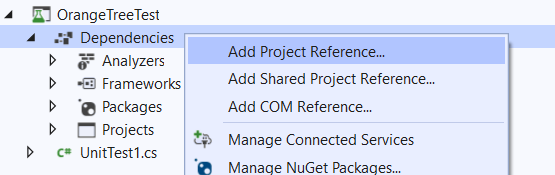
Hvis du er i tvivl, så se i Ex06-opgaven fra forrige uge, hvor du første gang også oprettede et testprojekt.

Men du er ikke helt færdig endnu. Lige nu kender testprojektet ikke til konsolprojektet, så det sikrer du ved at indsætte en projektreference i testprojektet til konsolprojektet:

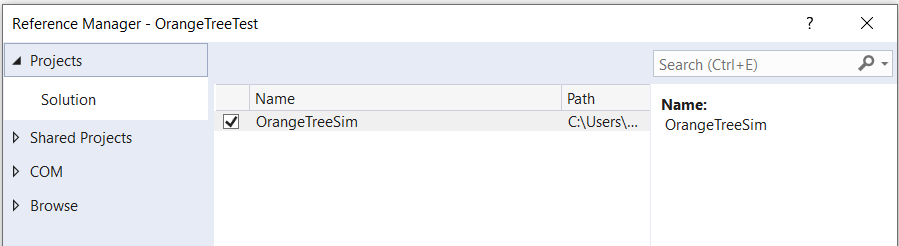
* Find “Dependencies” under testprojektet “OrangeTreeTest” i din Solution Explorer:



* Højreklik på ”Dependencies” og vælg ”Add Project Reference…” øverst på menuen.



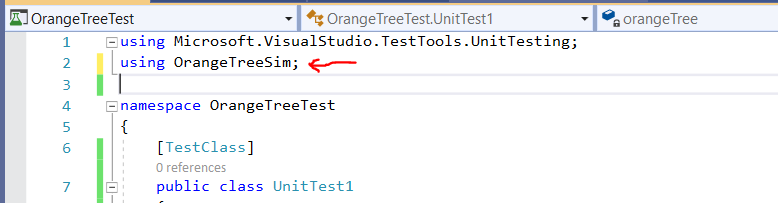
* Herefter sæt flueben i projektet ”OrangeTreeSim”, og vælg ok.



Det var projektreferencen.

Sidste trin er, at testprojektet også kender til ”OrangeTreeSim”-namespacet for at kunne tilgå klasserne i konsolprojektet:

* Øverst i testprojektets testfil ”*UnitTest1.cs*”, indsæt ”using OrangeTreeSim;” (se rød pil)



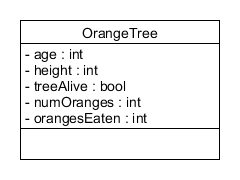
Nu kender testprojektet både konsolprojektet og det rette namespace, så det er parat til den sidste øvelse. Det kommer du til lidt senere.

## Øvelse 2.2: OrangeTree-klassen

Du skal nu fokusere på selve OrangeTree-klassens opbygning.

* Indsæt en ny C# klasse ”OrangeTree” (i en separat C#-fil) til konsolprojektet,
  + Gør klassen **public**

C#-klassen skal udtrykke følgende designklasse:



Klassen angiver et antal attributter, som samlet udgør klassens interne datastruktur. Bemærk, at alle attributter er angivet som private i klassen (minustegnet) og starter med lille bogstav (camel-case). Dette skal samlet indikere, at hver attribut skal implementeres som et **privat felt** i C#. Hver attribut ønskes at være privat for at beskytte klassens interne datastruktur fra misbrug af klassens brugere. Dette er en konkret måde at implementere OOP-princippet **indkapsling** på, som du også tidligere har set på.

Et instantieret objekt af OrangeTree-klassen repræsenterer således et konkret appelsintræ med sin helt egen tilstand (værdien af de fem attributter). Attributternes betydning er følgende:

* **age**: angiver træets alder
* **height**: angiver træets højde
* **treeAlive**: angiver (med sand/falsk), om træet er i live
* **numOranges**: antal appelsiner på træet
* **orangesEaten**: antal spiste appelsiner fra træet

Udfør følgende:

* Implementér de fem attributter i designklassen, som fem private felter i C#-klassen for OrangeTree.

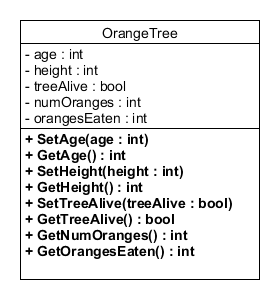
## Øvelse 2.3: Styret tilgang til datastruktur

OrangeTree-klassens datastruktur (attributer) er beskyttet med vilje, men du ønsker faktisk at kunne læse værdierne af alle attributter samt sætte værdien på tre af dem ude fra klassen. Men du ønsker også, at denne tilgang er kontrolleret uden mulighed for at ændre værdierne direkte. Da felterne i C#-klassen er private, kan du kun tilgå dem via nye offentlige (public) metoder til klassen, som netop kan sætte og læse værdier mere kontrolleret.

Ønsket er at kunne:

* **skrive og læse** age-feltet
* **skrive og læse** height-feltet
* **skrive og læse** treeAlive-feltet
* **læse** numOranges-feltet
* **læse** orangesEaten-feltet

Derfor opdateres designklassen med følgende operationer (ændringer angivet med fed skrift):



Bemærk, at disse operationer navngivningsmæssigt følger et bestemt mønster GetXXX/SetXXX, hvilket du snart i en senere opgave kommer til at arbejde videre med.

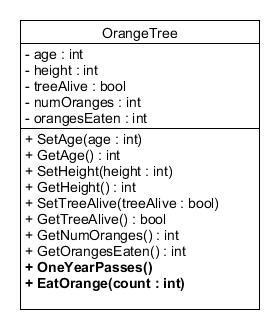
Udfør følgende:

* Implementér de angivne operationer som C#-metoder i OrangeTree-klassen, hvor en SetXXX-metode sætter en værdi i det tilsvarende XXX-felt, og en GetXXX henter en værdi fra det tilsvarende XXX-felt (f.eks. SetAge sætter age-feltet i klassen til værdien af parameteren age).
  + Hint: Du får behov for et ”access keyword”

## Øvelse 2.4: Styring af simuleringen

Nu, hvor du har datastrukturen på plads, skal du se på operationer til at simulere appelsintræets livsforløb. Dette gøres ved at implementere to nye operationer, der interagerer med træets tilstand.

Designklassen opdateres til følgende:



De to nye operationer og deres funktioner er:

* **OneYearPasses()**: denne operation simulerer, at et år er gået, træet er vokset og har givet ny frugt. For hvert år (dvs. et kald af OneYearPasses) ændres træets tilstand således:
  + Træets alder øges med 1 (age-feltet)
  + Træet højde øges med 2, hvis træets alder er under 80, ellers ikke (height-feltet)
  + Træet er i live, hvis træets alder er under 80, ellers dødt (treeAlive)
  + Træet giver ikke frugt første år, men derefter **øges** dets produktion med 5 appelsiner for hvert år (dvs. 5 appelsiner 2. år, 10 appelsiner det 3. år, 15 appelsiner det 4. år osv.). Frugt fra forrige år føres ikke med over til et nyt år (numOranges-feltet)
* **EatOrange(count : int)**: denne operationer simulerer, at der spises det antal appelsiner, der er angivet i count-parameteren fra årets høst, hvis der altså er så mange appelsiner tilbage på træet (orangesEaten-feltet)

Udfør følgende:

* Implementér de to operationer som metoder i C#-klassen

## Øvelse 2.5: Test af OrangeTreeSim

Du er nu parat til at teste din implementering.

Benyt ’git pull’ i materialefolderen for programmering. Herefter skulle folderen gerne indeholde word-filen: ”*Ex09-OrangeTree.UnitTest.docx*”.

Åbn “Ex09-OrangeTree.UnitTest.docx”, og kopier indholdet (alt med grå baggrund) fra filen ind i din UnitTest1.cs-fil i dit Unit Test projekt fra øvelse 2.1. Vær opmærksom på, at du kun overskriver TestClass med tilhørende TestMethods.

Bemærk, at der i starten af test-filen er en ny sektion:

[TestInitialize]

public void SetupForTest()

{

// Start an orange tree

orangeTree = new OrangeTree();

orangeTree.SetAge(0);

orangeTree.SetHeight(6);

orangeTree.SetTreeAlive(true);

}

Denne [TestInitialize] sektion udføres før hver enkelt test, og det er en smart og hurtig måde at instantiere og initialisere orangeTree-objektet uden at skulle skrive det samme i hver enkel testmetode.

Nu er du endelig klar til at benytte Test Exploreren:

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

Hvis du er heldig, vil alle dine tests være grønne, men sandsynligvis er der nok også røde. Gå hver testmetode igennem fra toppen af (hvis den fejler) og tilret, indtil den kører og er grøn.