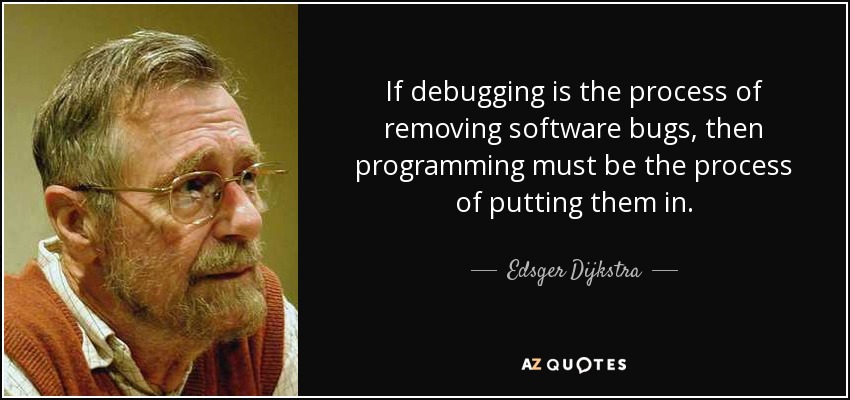
# Disaheim

|  |  |
| --- | --- |
| **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 huske enkelte kendetegn ved styring af **program flow** * [Unistrukturel] Du kan huske enkelte kendetegn ved styring af **datahåndtering** * [Unistrukturel] Du kan genkende enkelte kendetegn ved simple **C# datatyper** - int, string, bool, double, char, array (Sort), *class*, DateTime, enum * [Unistrukturel] De kan beskrive flere kendetegn i forhold til **OOP princippet indkapsling**   Færdighedssporet:   * [Unistrukturel] Du kan nævne enkelte kendetegn ved **testing** i Visual Studio IDE |
| **Din forberedelse** | Programmeringssporet:   * Enumerations:   + [C# Enum](https://www.w3schools.com/cs/cs_enums.asp)   + [C# - Simple Enumeration Example](https://www.youtube.com/watch?v=hK4-vOBFob0) (video: kun indtil 3:53)   + [YB] Afsnit 4.2 ”Enumerated Types” |

Du skal i denne (og de næste opgaver) arbejde videre med Disaheim-casen, som du kender fra systemudvikling. Udover at du introduceres for den sidste simple datatype, skal du primært arbejde med alle tidligere emner.

**Sørg for at få løst øvelse 2 i opgaven, da du skal arbejde med Disaheim-casen i næste opgave Ex16-Disaheim2.**

# Dagens ord



# Øvelse 1: Terminologi

Del teamet op i to mindre grupper, og brug **Møde på midten** i hver gruppe til at diskutere begrebet ”enumeration”. Husk, at alle i gruppen skal have taletid.

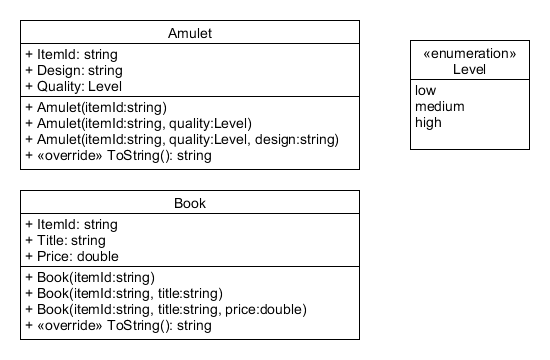
*Tidsramme: 20 minutter*

# Øvelse 2: Disaheim

Disaheim ønsker også at sælge bøger om wellness, meditation og levitation af uorganiske materialer. De har netop udvidet deres sortiment med sølvamuletter mod negative kosmiske stråler. Amuletterne er i flere design og forskellige kvaliteter (lav, middel og høj). Deres revisor tager sig af alt regnskab og betaling af moms og skat – det forstår Disaheim sig overhovedet ikke på og mangler derfor et overblik over deres forretning. De ønsker sig derfor en simpel applikation til at give dem dette.

Du skal anvende **parprogrammering** gennem hele opgaven. Du starter agilt i det små og bygger på efterhånden. Disaheim-casen vil blive udviklet over flere opgaver, så det er vigtigt, at du og din makker får løst den, da næste opgave er afhængig af denne. Du får brug for at refaktorere flere gange, så **sørg for, at dit projekt er lagt på GitLab**.

Du har allerede arbejdet med Disaheim-casen. Det har DesignAway-firmaet også og har i første omgang designet nogle få designklasser til DCD’en (designklassediagrammet for hele systemet), som du skal implementere i C#:



**Navnekonvention:**

* Klassens navn skal være i UpperCamelCase/PascalCase og i ental
* Attributter, der er public (+) og starter med et stort bogstav (UpperCamelCase/PascalCase), skal implementeres i C# som properties med stort startbogstav.
* Attributter, der er private (-) og starter med et lille bogstav (lowerCamelCase), skal implementeres i C# som et felt (en: field) med lille startbogstav. **Bemærk:** Hvis et privat C# felt fungerer som et ’backing field’ for en property, da skal feltnavn foranstilles med en underscore (\_).
* En operation med samme navn som klassen (og uden returtype) skal implementeres i C# som en constructor (dvs. stereotypen «constructor» udelades), ellers som en metode

**Denne navnekonvention benyttes fremover i alle kommende opgaver med mindre andet nævnes.**

## Øvelse 2.1: Implementér Disaheim-løsning

Udfør følgende:

* Opret et konsolprojekt med navnet ”Disaheim”.
* Implementér først Level-enumeration som en C# enum (i en separat C#-fil).
* Implementér dernæst de to designklasser Amulet og Book i hver sin C#-fil. Husk at gøre klasserne public.
  + Erklær klassernes properties (inkl. interne datastruktur)
  + Kontroller, at dine constructors følger mønsteret vist nederst side 106 i [YB] (afsnit 4.7.4) med constructor overloading og chaining
  + Kontroller også, om du har implementeret en ’no-arguments’ default constructor. Hvis du har, så skal du måske kigge på diagrammet en ekstra gang og overveje, om det er rigtigt.   
    Ret din kode om nødvendigt.
  + Implementér ToString()-metoden, som er en override.  
    Overvej, hvor ToString()-metoden kommer fra. Hvad er det, som bliver overrided?

## Øvelse 2.2: Videndel i teamet

Videndel (vælg selv CL-struktur):

* Sammenlign din og din makkers kode med de andre par i teamet med fokus på implementering af din enum-datatype, properties, constructor-koden, og ToString()-metoden.
* Ret i din kode, om nødvendigt.

## Øvelse 2.3: Testprojekt

Nu er det tid til en test. Udfør følgende:

* Opret et nyt testprojekt ved navn ”DisaheimTest”.
  + **Indsæt projektreference til dit projekt ”Disaheim” (husk også reference til namespace).**
* Erstat indholdet af UnitTest1 (dvs. kun testklassen) med unit-testen ”*Ex15-Disaheim.UnitTest.docx*”, der ligger i git materiale-mappen (brug git pull).   
  Husk at sætte testklassen **ind i** namespacet’s kodeblok.
* Få testen til at køre succesfuldt. Ret din kode om nødvendigt.

## Øvelse 2.4: Debugging

Debug mindst én af testmetoderne i unit-testen:

* Indsæt et breakpoint (F9: Toggle Breakpoint) i starten af Init()-metoden i UnitTest1.cs (dvs. i [TestInitialize]-sektionen).
* Indsæt også et breakpoint (F9) i starten af den udvalgte testmetoden.
* I Test Explorer vælg testmetoden, højreklik den, og vælg Debug-kommandoen.
* Brug Step Into (F11) og Step Over (F10) til at følge din kode trin for trin først i Init()-metoden og derefter i din udvalgte testmetode samt inspicér dine variable, dvs. parametre, properties, felter og lokale variable.
  + Observér hvordan dine constructors kalder hinanden, overfører parametre og sætter dine properties.

# Øvelse 3: Individuelt: Løs tidligere opgaver

Hvis du mangler at få løst nogle af dine tidligere opgaver, så tag fat i dem i den resterende tid. Arbejd meget gerne individuelt, men hvis der er en makker i dit team, som er samme sted som dig, arbejd gerne sammen via parprogrammering.