Opgaver til **Klasser**

1. Herunder har jeg skrevet en simpel klasse Person

```
class Person
{
    string Fornavn;
    string Efternavn;
    int Alder;
}
```

   a. Omskriv de to private instansvariable til offentlige properties med validering. Find selv på valideringskriterie.
   b. Tilføj to properties, **Far** og **Mor** på denne klasse. Hvilken type vælger du til Far og Mor? – tænk på, din mor og din far har samme egenskaber som dig, og har også forældre.
      i. Tilføj validering til disse properties.
   c. I din Main, lav en person, med dit navn, dine forældre, og deres forældre.
   d. Lav en ny klasse PersonPrinter
      i. Tilføj en metode til at printe en person
         1. f.eks. ”fornavn efternavn, alder”
      ii. Tilføj en metode, der printer alle navnene i stamtræet ud.
         1. (hint: speciel situation når vi ikke har angivet forældre (null))

2. Tilføj en ekstra constructor til klassen Person fra opgave 1, så man kan vælge at angive forældre når man konstruerer en instans af Person.

3. Tilføj en offentlig property PersonID til klassen Person fra opgave 1. PersonID er et ID der unikt identificerer personer. Brug din viden omkring statiske medlemmer til at implementere PersonID for Person-klassen, så alle instanser af Person har et unikt ID.
4. Lav et program der, for et bibliotek (folder/mappe) på din computer (f.eks. c:\)
   a. Udskriver alle filers navne og deres størrelser

b. Udskriver navnene på alle mapper og antal filer og mapper, de indeholder
   Brug klassen `DirectoryInfo`. Den ligger i namespacet `System.IO`. Der findes en tilsvarende `FileInfo` for filer.

5. I C# sammenligner vi værdier med ==. For de simple typer sammenligner vi værdier:

   1 == 1 giver true
   1 == 2 giver false
   true == true giver true
   true == false giver false

Når vi sammenligner referencetyper, er det, måske overraskende, ikke de data der refereres til der sammenlignes, men selve referencen!

a. Skriv et lille program der kan bekræfte dette. Lav ny klasse til formålet.

6. Definer en vektortype, om det er 2D eller 3D, er op til dig.
   a. Implementér metoder for addition og subtraktion
   b. Implementér skalering og evt. andre som f.eks. krydsprodukt.
   c. Valgte du at mutere dine objekter for vektorfunktionerne?
      i. Hvis dine vektorer muterer, implementer da en ikkemuterende version(returner altid nye vektorer). Og omvendt.