# .NET and C# Exercises dag 10

**1) Unit testing during development**
Make a project of type ClassLibrary with two classes.

An **Account**-class with two properties

int Balance with get
int MinBalance with get and set

and methods to put money onto the account and take money from the account

| | |
|---|---|
| void deposit(int amount) | |
| void withdraw(int amount) | If the new balance would be less than MinBalance, the balance must be unchanged, and an exception of type *ArgumentException* or you own defined Exception, *IllegalWithdrawException* must be thrown. |

A **Customer**-class with a List<Account> and methods

| | |
|---|---|
| void AddAccount(Account account) | an Account may only be added once |
| int TotalBalance() | |

a) Write the classes Account and Customer *with empty methods*.

b) Add a test project to your solution. Write testcode to test all what you think should be tested - remember Exception. Use descriptive name for a testmethod.
You may decide whether to use one or two testclasses.

c) Implement gradually the methods and test after each method is implemented.

**2) Yahtzee**
The application for this exercise is `dag10opg2App`; which uses the class library `dag10opg2_lib`. You are only going to expand the classlibrary, not the Application.

Run the application `dag10opg2App`:

You roll the Dices pressing "Roll"-button, can hold some of the dices (or all). After three rolls you choose a result by double-clicking a yellow TextBox.

It is your task to implement 5 methods in the Yahtzee class in `dag10opg2_lib`, to compute the value of 5 combinations based on what the dices shown. These methods are marked with a comment containing the word "missing"; they are at the end of the class. *Before* you implement one of these methods create a testProject and implement testMethods for the method.

You can read about the Yahtzee game and scores in

http://en.wikipedia.org/wiki/Yahtzee

or the Danish version

http://da.wikipedia.org/wiki/Yatzy

## Detaljeret forklaring til Yatzee klassen

Yahtzee klassen kan med en metode der kan kaste terningerne, dvs. give dem en tilfældig visning. Dette er naturligvis fuldstændig ubrugeligt til test. I stedet må vi under testen sørge for at terningen har den visning vi ønsker at teste, eller rettere test mange kombinationer af visning af terningerne. Yahtzee klassen har enkelte erklæringer som er af betydning for UnitTesting.

```
public class Yahtzee
{
    Random random = new Random();

    int[] dies = new int[5];
    bool[] hold = new bool[5];
    int[] stat = new int[7];
```

Terningerns visning findes i dies[] arrayet, hvorimod stat[] arrayet indeholder optælling af antal 1'ere, antal 2'ere etc.

Viser terningerne f.eks. 4 2 6 4 5 er indholdet af dies og stat følgende

**dies**

| Index | **Value** |
|---|---|
| 0 | **4** |
| 1 | **2** |
| 2 | **6** |
| 3 | **4** |
| 4 | **5** |

**stat**

| Index | **Value** | |
|---|---|---|
| 0 | **-** | |
| 1 | **0** | |
| 2 | **1** | |
| 3 | **0** | |
| 4 | **2** | |
| 5 | **1** | |
| 6 | **1** | |

Når du vil teste om en metode giver den korrekte værdi, f.eks. metoden valueTwoPair(), skal det foregår således:

- dies arrayet sættes til ønsket kombination
- stat arrayet sættes tilsvarende

begge dele ved at benytte PrivateObject classen.

Derefter benyttes Assert på den forventede værdi og værdien som valueTwoPair returnere.
Du skal benytte PrivateObject for at kunne initialisere de to arrays.

Hvis du i en testmetode kun benytter et af disse arrays er det selvfølgelig nok at initialiserer dette array.

**3) Designe til testbarhed**
Du skal arbejde videre med projektet `dag10opg3_start` i solution `dag10demo`, klassebiblioteket er inkluderet i projektet (MyClasses).

Det er starten på en WPF applikation der skal arbejde med ansøgninger til et erhvervsakademi. Foreløbig er der kun programmeret, den del der skal initialisere en ComboBox via Binding til en liste af Semester objekter, samt vælge det semester der skal arbejdes med.

Valg af den rigtige SelectedValue, når programmet starter er bestemt af dagen hvor programmet åbnes. SelectedValue har værdier som `'F07', 'E07', 'F08', 'E08', ... , 'E20'.`

Den rigtige SelectedValue beregnes ud fra aktuel dato efter følgende regel:

a) Mellem 1. januar og 15. februar 2013 vælges aktuel semester til ”E12” (Efterårssemester 2012)
b) Mellem 16. februar og 30. september 2013 vælges aktuel semester til ”F13” (Forårssemester 2013)
c) Mellem 1. oktober og 31. december 2013 vælges aktuel semester til ”F14” (Forårssemester 2014)

Både initialisering af semesterlisten og valg af aktuel semester foregår i Windows_Loaded eventhandler’en.

Sæt dig godt ind i koden i MainWindow.XAML.cs og MyClasses.cs.

Kør evt. programmet.

Afgør i hvor høj grad programmet er velegnet til test eller ikke egnet til unitest.

Denne opgave går ud på, at redesigne programmet samt at lave UnitTest af initialisering af SemesterList klassen samt valg af aktuel semester:

1) Test koden som vælge aktuelt semester som beskrevet ovenfor i a)-c)
2) Test at semesterlisten mht. antallet af semestre (28), og at første semesters Init er ”F07” og sidste semesters Init er ”E20”; dvs. du behøver ikke teste teksterne ”Forårssemester 20nn” og ”Efterårssemester 20nn”.

**4) Brug af Binding for at øge testbarhed**
Du skal udvikle unittest på projektet `dag10opg4_lib`, som benyttes af projektet `dag10opg4App`.

Det er detaljer fra en WPF applikation, der skal arbejde med ansøgninger til et erhvervsakademi.

Kør programmet:
Der kan indtastes eksamensår og eksamensgennemsnit samt året for studiestart.
Et omregnet eksamensgennemsnit beregnes efter følgende regel: Hvis eksamensåret højst er 2 år før året for studiestart og eksamensgennemsnittet er mindst 8.0, så er ganges eksamensgennemsnittet med 1.08 (og afrundes til 1 decimal), i alle andre tilfælde er det omregnede gennemsnit lig med eksamensgennemsnittet.

Prøv at ændre på værdien af de tre øverste TextBox'e og se at omregnet gennemsnit ændres i overensstemmelse med ovenstående regel.

For at gøre Applikationen let at teste er næsten al kode flyttet væk fra code behind filen ved at benytte databinding:

- En klasse SemesterData indehold alle de tal der vises i GUI'en i fire properties, EksAar, EksSnit, StudieAar og OmregnetSnit.
- Klassen implementerer INotifyPropertyChanged for at GUI'en kan få besked, når en property er ændres:
    - Når en property i et object af typen SemesterData ændres fyres eventen PropertyChanged i objektet. Det betyder at GUI'en "får besked" når en property i objektet ændres.
- I XAML koden definere Binding til de fire TextBox'e.
- I Windows_Loaded eventhandleren sættes DataContext for Window til en instans af SemesterData.

De sidste to punkter sikrer at GUI'en og SemesterData er synkroniseret.

Applikationen er programmeret færdig, men ikke testet med unittest. I denne skal du skrive unittest til applikationen *ved at teste et SemesterData objekt*:

a) Test at omregnet gennemsnit beregnes korrekt i alle tilfælde.

b) Udtænkt en måde hvorpå du i dit UnitTest projekt kan kontrollere at PropertyChanged eventen i SemesterData virker som den skal, dvs. fyres når en værdi ændres og IKKE fyres når en property sættes til den værdi, den i forvejen har.

**Hjælp:** Hver opmærksom på at ændring af f.eks. EksAar kan resultere i at PropertyChanged fyres to gange først når EksAar ændres og dernæst når OmregnetSnit ændres (nemlig hvis reglen om omregnet gennemsnit ganger eksamensgennemsnit med 1.08.)