Unit test opgave

# Opgave 1

Formål: at arbejde med unit test

Vi skal arbejde videre med bil opgaven som vi så på sidste gang. Du/I kan arbejde videre på jeres egen eller downloade løsningen fra forrige gang til at arbejde videre op.

Tilføj et test project til solution’en.

Lav en ny testmetode for hver test og giv metoden et sigende navn, som fortæller hvad der testes i metoden fx. TestDieselBilGrønafgift og TestDieselBilRegAfgift2016 og brug ikke TestMehod1 , TestMethod2.

Inden du går igang med at skrive testen kan det være en god ide at regne ud hvad testen skal give af resultat så gør det først.

**Tip: højreklik på klassenavnet for den klasse du vil teste og vælg “Create Unit test” så vil der blive genereret default testmetoder.**

1. Test at registreringsafgiften udregnes rigtig for alle biltyper , husk at det er vigtigt at test grænsetilfældende, så lav test med årgang 2015 og 2016 samt hvor prisen ligger lige på 80500 i 2015 og prisen er 81700 i 2016.
2. test grøn afgift

**Grønafgift**:

DieselBil = 3000 kr.

BenzinBil= 1500 kr.

ElBil = 0 kr.

1. test at rækkevidden regner rigtigt

Eksempel på testmetode

|  |
| --- |
| /// <summary>  /// tester at grønafgift beregnes rigtigt for ElBil  /// </summary>  [TestMethod]  public void TestElBilGrønAfgift()  {  ElBil bil = new ElBil(80500,2015,20,30);  Assert.AreEqual(bil.GrønAfgift(), 0);  } |

# Opgave 2

Test at de rigtige exception bliver kastet.

Først skal vi lige kaste nogle exceptions hvis input data ikke er som vi forventer

I metoden Registreringsafgift() :

1. Hvis prisen på bilen <= 0 skal der kaste en ArgumentException() samt en beskrivende tekst til fejlen fx “"prisen må ikke være <= 0"”
2. Hvis købsår er < 2014 skal der kastes en ArgumentException()

´Skriv test som er dækkende for at teste at ArgumentException bliver kastet

Over testmetoden tilføjes følgende attribut

[ExpectedException(typeof(ArgumentException))]

Husk igen at det er vigtigt at teste grænserne.

# Opgave 3

For at undgå at vores program går ned skal vi implementerer en try - catch blok når vi kalder metoden Registreringsafgift(). I catch blokken kan vi udskrive fejlmeddelesen til konsollen

# Opgave 4

Vi skal nu prøve metoden Test Driven Development, det vil sige at vi skriver testene først og derefter skriver koden til programmet.

Vi skal beregne GrønAfgift på Dieselbiler ud fra hvor mange km/l den kører. Brug nedenstående information .

5-15 km 8000 kr

>15 -25 km 4000 kr.

>25 km 1500 kr.

Hvis dieselbilen IKKE har dieselpartikelfilter koster det 1000 kr ekstra

Hvis Km/l er mindre end 5 km/l skal der kastes en argumentException

Du skal nu skrive de test som er dækkende for GrønAfgift på Dieselbil. Hver gang du har kodet en del af løsningen kan du nu teste op imod dine unit test og se om du har kodet rigtigt.

# 

# Opgave 5

Kod din egen exception fx. GrønAfgiftException, skal nedarve fra Exception

Kast denne exception i metoden grøn afgift istedet for argumentException’en hvis Km/l mindre end 5 km/l.

|  |
| --- |
| public class GrønAfgiftException : Exception  {  const string **text** = "Grønafgift er forkert,Km/l er mindre end 5 km/l ";  public GrønAfgiftException()  :this(**text**)  {  }  public GrønAfgiftException(string message)  : base(message)  {  }  public GrønAfgiftException(string message, Exception inner)  : base(message, inner)  {  }    } |

©Martin Kierkegaard