**Øvelser i Method Extension og Events**

Dette er dagens øvelser i Extension Methods og i Event.

Nogle øvelser kræver at i opretter unit test som i forrige uge.

God arbejdslyst.

**Øvelse 1 - Udvidelse af string-klassen (extension-metode)**

Af ukendte årsager ønsker du at udvide string-klassen med en ekstra metode Capitalize(), som returnerer en ny tekststreng, hvor det første tegn i tekststrengen er ændret til stort bogstav (Uppercase).

Da du ikke kan ændre implementeringen af selve string-klassen, må du gøre det ved at at lave en extension-metode til klassen.

Implementér Capitalize() som extension-metode

Udfør følgende:

* + Opret et nyt konsolprojekt i Visual Studio med navnet ”StringExtensions”
  + Opret en ny klassefil ”Extensions.cs” i projektet
  + Implementér extension-metoden Capitalize() uden input-parametre.

**Øvelse 2 - Test af Capitalize()-metoden**

Udfør følgende:

* + Opret derefter et testprojekt i løsningen.
  + Indsæt testen angivet i nederste tekstblok.
  + Kør testen, og ret fejl om nødvendigt.

    [TestClass]

    public class UnitTest1

    {

        [TestMethod]

        public void TestCapitalize1()

        {

            // Arrange

            string s = null;

            // Act

            string s2 = s.Capitalize();

            // Assert

            Assert.AreEqual(null, s2);

        }

        [TestMethod]

        public void TestCapitalize2()

        {

            // Arrange

            string s = "";

            // Act

            string s2 = s.Capitalize();

            // Assert

            Assert.AreEqual("", s2);

        }

        [TestMethod]

        public void TestCapitalize3()

        {

            // Arrange

            string s = "opgave";

            // Act

            string s2 = s.Capitalize();

            // Assert

            Assert.AreEqual("Opgave", s2);

        }

        [TestMethod]

        public void TestCapitalize4()

        {

            // Arrange

            string s = "Opgave";

            // Act

            string s2 = s.Capitalize();

            // Assert

            Assert.AreEqual("Opgave", s2);

        }

        [TestMethod]

        public void TestCapitalize5()

        {

            // Arrange

            string s = "o";

            // Act

            string s2 = s.Capitalize();

            // Assert

            Assert.AreEqual("O", s2);

        }

        [TestMethod]

        public void TestCapitalize6()

        {

            // Arrange

            string s = "O";

            // Act

            string s2 = s.Capitalize();

            // Assert

            Assert.AreEqual("O", s2);

        }

    }

**Øvelse 3 - Random**

I C# repræsenterer Random-klassen en generator af tilfældige tal. Random-klassen giver metoder til at generere tilfældige heltal, flydende punkt tal og andre tilfældighedsrelaterede operationer.

Random-klassen indeholder en metode: ”NextDouble()”som returnere tilfældig en værdi mellem 0 og 1.

Af uransagelige grunde ønsker du at udvide Random-klassen med en ekstra metode som returnerer en værdi mellem 0 og en given maks værdi.

Du kan læse mere om Random-klassen her: <https://code-maze.com/csharp-random-class/>

**Øvelse 3.1 - NextDouble()**

Udfør følgende:

* + Implementér extension-metoden NextDouble(), som skal kunne kaldes med én parameter af typen double, der angiver den maksimale værdi for metoden skal returnere.

**Øvelse 3.2 - Plat eller krone**

Du ønsker nu at simulere ”plat eller krone”, hvor en tilfældig boolsk værdi bliver valgt. Møntkastet skal være fair dvs. at 50% af gangene skal det være plat og de resterende 50% krone. Af og til vil vi gerne have muligheden for at ændre sandsynligheden så vi får en skæv fordeling (f.eks. 75% true og 25% false).

Udfør følgende:

* + Implementér extension-metoden CoinFlip(), som tilfældigt vælger en boolsk værdi.
  + Metoden skal have en valgfri parameter, der indikerer frekvensen af hvor ofte man slår plat.
  + Hvis man ikke angiver en værdi i sin parameter, så skal dens default være 0.5 eller 50%.

**Øvelse 3.3 - Random string**

Du skal nu oprette endnu en extension metode til random klassen. Metoden skal tilfældigt kunne vælge en af flere strings, som f.eks. ”Nord”, ”Syd”, ”Øst” og ”Vest”, hvor hver string har samme sandsynlighed for at blive valgt.

Udfør følgende:

* + Implementér extension-metoden NextString().
  + Metoden skal have et array af strings som parameter.
  + Metoden skal returnere en given string fra arrayet, der hver gang tilfældigt bliver valgt.

**Øvelser med Events**

De næste øvelser knytter sig til Events.

Genbesøg eventuelt læringsobjektet om Events

**Øvelse 4 - Sparegris**

Implementér en simpel klasse, som repræsenterer en sparegris.

I et loop skal det være muligt at lægge et beløb i grisen (InsertAmount(double amount)). Loopet skal fortsætte, indtil der indsættes 0 (nul). InsertAmount skal rejse et event, som skal bevirke at:

Sparegrisens saldo udskrives på skærmen:

* + Når sparegrisens saldo overstiger 399, skal der udskrives ”Tillykke – du har nu penge nok til et sæsonkort til OBs kampe”
  + Når sparegrisens saldo overstiger 799, skal der udskrives ”Tillykke – du har nu penge nok til et premium sæsonkort til OBs kampe”
  + Når sparegrisens saldo overstiger 1197, skal der yderligere udskrives ”Tillykke – du har nu penge nok til tre sæsonkort til OBs kampe”
  + … og så fremdeles

**3 sekunders pause!**

En meget nem måde at holde tre sekunders pause i et program er ved at anvende følgende udtryk:

**System.Threading.Thread.Sleep(3000);**

NB! Vi har om threads senere på dette semester

**Øvelse 5 - Metronom**

En simpel *Metronome* klasse skal rejse et event, *Tick*, hvert tredje sekund.

Dette skal gøres i et loop, som kører i alt 5 iterationer.

En anden klasse, *TickListener*, skal definere en metode, som skriver ”Heard you” til skærmen for hvert ”Tick”.

**Øvelse 5.1 - Implementér med en delegate**

Design først din løsning, hvor du bruger en delegate til din Tick-metode. Brug evt. klassediagram og SD for at få overblik. Vær omhyggelig med, hvordan du definerer din delegate og den tilhørende metode samt hvordan du binder det hele sammen. Du behøver ikke kunne tegne det helt korrekt – brug blot diagrammer for at få overblikket.

Udfør følgende:

* + Implementér Metronome- og TickListener-klassen.
  + Skriv de nødvendige sætninger i Main() (i Program-klassen) til at afprøve de ovenstående to klasser
  + Kør programmet, og tilret, indtil det virker.
* **Øvelse 5.2 - Implementér Metronome-klassen med et event**

Udfør nu følgende:

* + Omdan din Tick delegate til et event i stedet
  + Hvis din kode evt. ikke kan kompileres, da overvej, hvad der kan være galt og tilret.
  + Kør koden.

Overvej nu til sidst, hvad forskellen er på en delegate og et event, og hvorfor man vil vælge at implementere et event og ikke kun en delegate.