Day 11 .NET and C# Exercises: Threading

1) *Meget simpel opgave*
Lav en Console Application med følgende to klasser, Program og Worker

```
class Program
{
  static void Main(string[] args)
  {
    Worker worker = new Worker();
    Thread t1 = new Thread(new ThreadStart(worker.work)); t1.Name = "Th1";
    Thread t2 = new Thread(new ThreadStart(worker.work)); t2.Name = "Th2";
    Thread t3 = new Thread(new ThreadStart(worker.work)); t3.Name = "Th3";

    t1.Start();
    t2.Start();
    t3.Start();

    Console.ReadLine();
  }
}

public class Worker
{
  public void work()
  {
    for (int i = 1; i <= 40; i++)
    {
      // udskriv en linje
      Console.Write("Output from");
      Console.Write(Thread.CurrentThread.Name);
      Console.Write(": " + i);
      Console.WriteLine();
    }
  }
}
```

Kør programmet og se en linje fra én tråd blandes med en linje fra en anden tråd.

Tilføj synkronisering således at trådene skriver en hel linje færdig inden en anden tråd skrive en linje:
a) med lock keyword'et
b) med Monitor

2) *Begrænset buffer*
Åben projektet `thread_opg2_start` i solution `dag11demo`

Kør programmet: Det har 2 producere og en consumer som henholdsvis lægger tilfældige string i en buffer og henter string i bufferen. (Producerne producerer 40 string, consumeren henter 80 string). Kig godt på koden, så du får overblik over den.

a) Eksperimentér med programmet:
- Lav en ny consumer, ret begge så de hver henter 40 string i buffer)
- Ret det antal millisekunder som comsumer venter mellem hver hent; prøv både mindre og større delay.

b) Gå tilbage til oprindelige version. Lav en udvidelse så bufferen maksimalt vil indeholde 5 string.

3) *Scanning for txt-files with a BackgroundWorker*

Program a WPF applikation with a GUI as shown below

Write a recursive method

```
public void Scan(string path)
```

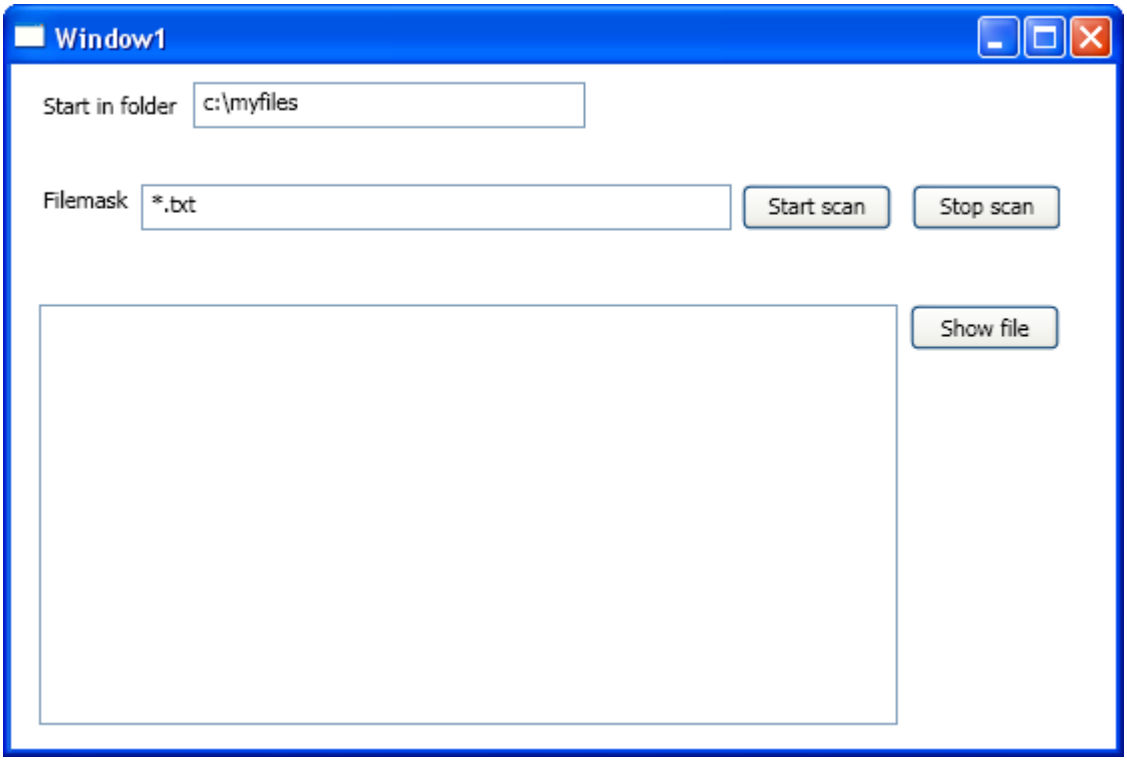
which is scanning f.eks. c:\myfiles searching for `*.txt` files (or other text files). The full path of the found files are added to the ListBox.

The scanning must be done in a BackGroundWorker thread, not to lock the GUI for a long time. But the WPF-window must be updated as soon a new files is found.

During the scan the user can open a found text file by the Button ”Show file”. The files is shown in another window.

```
// find all directories in c:\
string[] dirs=Directory.GetDirectories(@"c:\","*");

// find all *.doc files in c:\myfiles
string[] files Directory.GetFiles(@"c:\myfiles","*.doc");
```



Avoid scanning the two folders

`c:\WINDOWS` (or whatever your Windows folder is called)

```
c:\System Volume Information
```

which is prohibited (at least in previous versions of .NET).

Hjælp:
a) DoWork eventhandleren kalder en rekursiv metode, Scan, som scanner en mappestruktur.

b) Når du finder en fil i Scan-metoden, kalder du ReportProgress i BackGroundWork'eren:

```
worker.ReportProgress(0, filename);
```

hvor filename er en string med den fundne fil.

c) i ProgressChanged eventhandleren:

```
string filename=(string)e.UserState;
lBoxFiles.Items.add(filename);
```

d) Du åben en fil med standard programmet i Windows til den type filer ved at kalde

```
Process.Start(filename);
```

4) *Binding*
Omskriv opgave 3 så den i stedet benytter binding.

Måske er det lettest hvis du laver et nyt WPF projekt;

- fra XAML filen kopierer det indre af <Window>-tagget, men ikke <Window>…</Window> selv (det indeholder namespace).
- Kopier også Scan metode – eller forhåbentlig filen med Scan metoden – til det nye projekt. Så har du hele brugergrænsefladen og Scan-metoden.

Find ud af hvordan du vil afbryde Scan, når brugeren rykker på "Stop scan" knappen.

HUSK:
`BindingOperations.EnableCollectionSynchronization(…, …);`