Horse Race

Oplevering Threading in C#

Versie 1.0
02 April 2020

Docenten:
Rob Loves
Rob Smit

Groep:
Yaron Lambers - 1569488
Thomas Christerus - 1550965
Lennart Pikijn - 1576565
Michiel Jansen - 1571997

Inhoudsopgave

Moduledoelstellingen

Voor ons programma hebben wij de volgende begrippen uit figuur 1 toegepast. De codesnippets komen uit het project, dat te vinden is op https://github.com/Michieljansen1/HorseRaceThreading.

- ASync/Await;
    - Toegepast op veel plekken onder anderen in de class **PageLoader.cs**. Async is onder andere toegepast omdat de Sockets class gebruik maakt van een Async methode omdat deze wacht op resultaat van een ingevoerde website. Een voorbeeld hiervan is:

```
await socket.ConnectAsync(new HostName(_url), "80");
```

- Locking (interlocking);
    - In de class **Ping.cs** wordt de interlocking gebruikt om de score goed te kunnen updaten. Zonder deze lock worden bepaalde scores overschreven. Dit komt omdat meerdere threads tegelijkertijd de variabele willen bijwerken. De eerste parameter is de bij te werken variabele. Een voorbeeld van gebruik van de lock is:

```
Interlocked.Add(ref _totalTime, ms + delta);
```

- Tasks
    - In de class **PageLoader.cs** wordt gebruik gemaakt van tasks, omdat er content van meerdere webpagina's geanalyseerd wordt. Hierdoor worden deze pagina's gelijktijdig in behandeling genomen. Een voorbeeld hiervan is de onderstaande code, waarin de volledige grootte van alle opgehaalde pagina’s gelijktijdig wordt berekend en dit terug geeft.

```
public async Task<int> Run(string url)
{
   foreach (var page in pages)
   {
       _totalSize += await GetDomSize(url+page);
   }

   return _totalSize;
}
```

- Threadpool
    - in de class **Ping.cs** wordt een Threadpool gebruikt, zodat deze verschillende threads kan afhandelen. Omdat iedere ping een andere responstijd kan hebben, kunnen deze acties zonder specifieke volgorde uitgevoerd worden. Een voorbeeld hiervan is:

```
_pingAction = Windows.System.Threading.ThreadPool.RunAsync(async (workItem)
```

- LINQ
    - Linq is gebruikt voor het makkelijker af te handelen van lijsten. Door Linq te gebruiken kunnen wij ook beter filteren, groeperen en sorteren van data . Een voorbeeld hiervan is:

```
var horse = from horse in Horses where horse.distance = 10
```

1.4 Module doelstellingen

Na het succesvol afronden van deze module kan de student:

- De volgende begrippen uitleggen en/of toepassen:
    a. (Multi)Threading;
    b. Locking;
    c. Mutex;
    d. Semaphore;
    e. Threadpool;
    f. Delegates;
    g. Tasks & Multitasking;
    h. Async/Await;
    i. LINQ/PLINQ;
    j. Asychronous I/O.
- Vanuit Microsoft Visual Studio 2015 of nieuwer:
    a. Een klassendiagram maken;
    b. Werkende code genereren vanuit het klassendiagram;
- Een multithreaded systeem ontwerpen en bouwen welke over verschillende processorcores communiceert door middel van C#.

Figuur 1, Doelstellingen

Code of conduct

De class opbouw moet worden gehanteerd zoals is aangegeven in dit document. Aanvullende informatie is te vinden via deze link.

Class opbouw

```
    /// <summary>
    ///
    /// </summary>
public class TestClass
{
    private int _count;                 // Description variable
    private String _name            // Description variable

    /// <summary>
    ///
    /// </summary>

    public TestClass(String name)
    {
        // Setting class attributes
        _name = name;
        _count = 0;
    }

    public String Name
    {
        get { return _name; }
        set { _name = value; }
    }
```

```
    /// <summary>
    ///
    /// </summary>
    /// <param name="name">param description</param>
    public void SetName(String name)
    {
        // Add 'is my name!' to the name variable
        name += " is my name!";

        // Set the name variable
        _name = name;
    }

     /// <summary>
    ///
    /// </summary>
    public void PublicTestMethod()
    {

    }



    /// <summary>
    ///
    /// </summary>
    protected private void ProtectedTestMethod()
    {

    }

     /// <summary>
     /// Description method
     /// </summary>
     /// <param name="count">param description</param>
    private String privateTestMethod(int count)
    {
        return "";
    }
}
```

Versiebeheer

Wij hebben gebruik gemaakt van de Git Flow waar twee beschermde branches zijn aangemaakt: master en develop. Vanuit de develop zijn telkens nieuwe featurebranches aangemaakt op basis van vooraf bepaalde issues. Na het afhandelen van een issue werd een pull request aangemaakt naar de develop branch. Voordat een pull request gemerged werd in de develop branch, werd er eerste een code review uitgevoerd door een ander groepslid. De contributions zijn geen goede peilers omdat voor complexe problemen gebruik is gemaakt van screenshare.

Grafische user interface

De grafische user interface (GUI) is gemaakt met behulp van een grid systeem. De GUI is responsive gemaakt en gebruikt worden op alle ondersteunde UWP platformen. Het design, te zien in figuur 2, is gemaakt aan de hand van het vooraf gemaakte UML.

Figuur 2, wireframe.

UML

Omdat er verschillende implementaties in de praktijk anders werkten dan aanvankelijk gedacht, zijn er wijzigingen gemaakt aan de UML. De C# Ping library is niet te gebruiken met UWP, hierdoor is eigen logica in de Ping class toegepast. In figuur 3 is de meest recente UML te zien. Deze correspondeert met de meest recente versie van de code.

FIGUUR 3, Nieuwe UML

L. Pikijn, Y. Lambers, T, Christerus, M. Jansen | April 2, 2020