C:\Users\Manfred Stueber\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Logo_Hauptcampus_Informatik.eps

Advanced Programming – Application Description

Tobias Forve

Matrikelnummer: S00205070

Betreuer: Prof. Dana ....

Trier, 17.5.2020

Predescription

Goal of the lecture was it to write an application that deals with typical multithreading features. To achieve this the application was written in CSharp using the System.Threading library.

Table of Contents

2. In general .……………………………………………………………………………….4

2.1 Used programs and libraries ………………………………………………...4

2.2 Windows Forms elements used …………………………..4

3. Umsetzung……………………………………………………………………………….6

3.1 Wang Tile……………………………………………………………………………..6

3.2 Berechnen des Index…………………………………………………………………..6

3.3 Umsetzung im Code…………………………………………………………………..7

4. Fazit……………………………………………………………………………………...9

2. In general

2.1 Used programs and libraries

Programs used are visual studio 19 with windows forms and .NET 4.7.2. Main additional Libraries needed to achieve the goals of the application are System.Threading, System.IO and System.IO.IsolatedStorage, as well as the Windows Forms Libraries (System.Windows.Forms).

2.2 Windows Forms elements used

The Application consists of a Form Form1.cs which holds the other GUI elements of the Windows Form, three labels label1 label2 and label3 just to label parts of the GUI. Two ComboBoxes, one called threads which holds options to choose from how to handle the main task of the application, the other called number that stores an integer which is used as a reference how many threads to create. It also hold a NumericUpDown timer to set up a integer between 0 and 2000 to set the seconds the Threads have to run. The last 2 elements are a button startStop to start the threads and a FlowLayoutPanel mainPanel to display important UI updates and processes.

3. Implementation

First task is to select one option in the ComboBox threads followed by the decision to choose a fix amount between one and four of threads to handle the task. In the NumericUpDown timer you have to decide how much time the threads have to process. Min time is 0 seconds max 2000. After put in all relevant data running the Start button will lead in to one of three possible options as followed.

3.1 “Threads” – Option 1

Choosing the first option “Threads” in the ComboBox will create lead to the WaitThread.cs Class, especially to the void Start() method in which one ParameterizedThreadStart will be raised calling the DoWork() method. In the DoWork() method an array of WaitHandle will be initialized with the amount of Threads to run to handle the synchronization of the upcoming Threads and handle shared resources.

Inside of the for-loop a new EventWaitHandle object called handle will be created with an initialState of false to handle further Thread queues.

After that the time to process will be divided by the amount of threads to split the time between them. Between one and four anonymous Threads will be raised to count down the time. handle.Set() set the EventWaitHandle state to signalized to give waiting threads the signal to also proceed.

With WaitHandle.WaitAll() it is insured that all WaitHandle objects signalized their status.

At the end the OnThreadEvent will raise an Event to set the status of ThreadeventArgs, Status 1 means everything worked fine and the threads have fnished, 2 means the threads were interrupted by the user clicking the Button again and calling the startStop\_Click method in Form1 and status 3 means an error was occurred.

3.2 “Threads Queue” – Option 2

Inser text here

3.3 “Background Worker” – Option 3

Insert text here