# Exercises about Asynchronous methods

* Solve them in Visual Studio.

## Exercise 20.01

|  |  |
| --- | --- |
|  |  |
|  | This exercise is based on 08.03 – Bubble sort and 08.04 – Insertion sort. |
|  |  |

* Create a WPF or a Windows Forms application.
* There are several buttons on that form.
* Button 1.
  + Generates an array of 100.000 (or more elements).
  + For testing purposes start with a smaller array.
* Button 2.
  + Create 2 copies of that array and starts Asynchronous the bubble sort on one copy and the insertion sort on the other copy.
  + When a sort routine is finished, the elapsed time is shown.
* Button 3.
  + Shows a Message Box on the screen where the bubble sort and insertion sort is at that moment.
* Button 4.
  + Cancels the sort operations.

### Variant 1

* Add two progress bars on the screen that shows the progress of both sort routines.

### Variant 2

|  |  |
| --- | --- |
|  |  |
|  | Pay attention.  The merge sort does not loop thru your array.  But divides it every time into 2. |
|  |  |

* Add a third sort routine to it. The merge sort, see exercise 08.05.
* So a third progress bar is needed.
  + You need to think how you are going to do this 😊.
* The button that shows a Message Box shows also on what level the merge sort is at that moment.

### Variant 3

* Calculate the speed differences between all the sort routines.

## Exercise 20.02

* Create a WPF or a Windows Forms application.
* There are 3 buttons on the screen.
  + The first button generates a file with a lot of numbers.
    - Every line contains one number.
    - This file can have a fixed name.
  + The second button opens a file open screen where I can give the path to another text file that contains numbers.
  + The third button reads the generated file (button 1) or the file that is given with the path (button 2).
    - It is the last button that was hit that decides what file is read.
    - All numbers in the file are added up.
* Show a kind of indicator (progress bar) that shows how long the program will be busy till everything is done.
  + So when the progress bar is at 100%, you show the result of the calculation.
* There is a cancel button on the screen.

|  |  |
| --- | --- |
|  |  |
|  | You can use 008 – AsyncCalculationFromDataFile.zip as starting point for this exercise.  Better is to study the example. And try to recreate it. |
|  |  |

## Exercise 20.03

* Create a C# console application with a method called “DownloadWebpageAsync” that accepts a URL as a parameter and returns a “Task<string>”.
* In the “DownloadWebpageAsync” method, use the “HttpClient” class to download the webpage at the specified URL and return the resulting HTML as a string.
* Create another method called “PrintWebpageAsync” that accepts a URL as a parameter and prints the downloaded webpage to the console.
* In the “PrintWebpageAsync” method, use await to call the “DownloadWebpageAsync” method and get the downloaded HTML.
* Call the “PrintWebpageAsync” method from the “main()” method with a URL of your choice.
* Run the application and ensure that the downloaded webpage is printed to the console correctly.
* Modify the “DownloadWebpageAsync” method to add a delay of 7 seconds before returning the downloaded HTML.
* Run the application again and ensure that the delay doesn't cause the application to hang, and that the downloaded webpage is still printed to the console correctly.

|  |  |
| --- | --- |
|  |  |
|  | You can use 007 – AsyncDataBound.zip as starting point for this exercise.  Better is to study the example. And try to recreate it. |
|  |  |