Log Task - Change Description

April 24, 2018

`AsyncLog` class

The following has changed in the `AsyncLog` class:

- Now uses a queue rather than a list as its internal buffer.
  - The queue used is thread safe to make sure that multiple threads can use it without running into issues.
- The code has been refactored to have less code duplication and tasks are delegated to smaller methods.
- The logger is programmed more defensively now to catch possible run-time exceptions.
- Comments and a bit of documentation have been added.
- The logger will stop buffering lines in the queue (i.e., when the write method is called it will not add them to the queue) when the logger has been asked to stop and flush. The logger stops when the queue is empty.
- The logger creates one file per day. It only appends the header the first time it is opened.
- The writer is closed when the logger stops.
- Logger construction does not end until main loop has writer lock (to prevent logger to write to a closed writer.).

`AsyncLog` class

- A minimum of tests has been created
  - A test for each of the requested features. (note: I am not sure how to test the immediate stop with unit tests. If one could actually take control of the scheduler, then one could make a deterministic test based on some scheduling. To my knowledge this is not possible which makes it impossible to make a unit test as such.)
  - The tests are called from Program.cs

- Introduced interface and class to get control over DateTime (that is introduced test stub).

Other comments

More things that could have been done in the project.

- I would have liked to write more unit tests for the smaller methods in AsyncLog to install more confidence in their correctness. I did, however, not get a unit test framework to work on my linux computer.
- Check the docs to make sure all run-time exceptions will be caught (one could make a temporary hack by simply catching all exceptions).
- The specification was vague, so the implementation could change in a number of ways depending on actual needs:
  - If the logger is used by multiple threads, and it is somewhat important to not through away log entries, then one could make the write method make sure that the line is written to the internal queue buffer.
  - The logger yields often to not slow down the main application. One could allow the logger to be more active (for instance by letting it write multiple entries before it yields. This could be done in the style of the original project with a counter that keeps track of the amount of lines written. When the limit is reached, the counter is reset and the logger yields).
  - The logger could allow for the user to actually pick the directory for logging.
- The documentation could be more thorough and utilise the available markup better.