PROG2121 – WINDOWS PROGRAMMING

A04 – TASKS

OVERVIEW

Many applications are composed of multiple threads to optimize performance. In most cases, this has the effect that the application can do several things at the same time.

Instead of Threads, you will use Tasks to develop a "multi-threaded" application that writes to a file. You will be monitoring the size of the file, and when it becomes a certain size, the application will close gracefully.

OBJECTIVES

- Implement a simple Task-based solution in a Windows application

ACADEMIC INTEGRITY AND LATE PENALTIES

- Please refer to the SET Policies document regarding Academic Integrity Information
- Please refer to the SET Policies document regarding Late Policy

EVALUATION

- Please see the rubric in eConestoga for details

REQUIREMENTS (MINIMUM)

Individual Submission

1. The application must be written in C#. You must write the application in a console application.
2. The must be 2 command line arguments supplied to the application:
    a. The filename of the file that must be written to
    b. The size of the file that will be created
    c. If the file already exists, you must alert the user and ask the user to confirm that the file will be overwritten. If confirmed, you must overwrite the file with new data. If the user does not confirm, the program ends.
    d. The minimum file size is 1,000 characters, and the maximum size is 20,000,000 characters. You must validate the argument. When validating the data, do not

use Exceptions for input validation. In case of an error, please notify the user appropriately.

   e. You must provide the "Usage" if the command line arguments are incorrect or invalid, or if the user enters the /? switch after the command as the first command line argument.

3. The application must use 25 identical Tasks to write random data to the single file the user specified at the start of the program. The random data must be in blocks of 36-character strings.
4. A *separate* Task must monitor the size of the file. At 0.1 second intervals, the size of the file is checked. If it is equal to or greater than the maximum file size supplied by the user at the start of the program, the writing Tasks must be stopped in a controlled manner. Please note that the final file size may be a little larger than the target size. This is acceptable since some Tasks may need to complete a write operation as part of the controlled stop.
5. As the program runs, do not show the status of the Tasks that are writing to the file.
6. As the program runs, display the size of the file at 0.1 second intervals.
7. When the file size is met or exceeded, display the final size of the file.
8. You must make use of the GitHub Classroom repository to make regular commits of your code.

Notes:

- The information on how to find the length of a file can be found here: Common I/O Tasks | Microsoft Docs
- All file operations must be enclosed in try-catch-finally blocks.
- There are many ways to solve this problem. You may extend the solution with additional Tasks, data structures, etc. However, the requirements above must be met.

FILE NAMING REQUIREMENTS

- There are no specific file naming requirements. Please keep the name short, but meaningful.
- The submitted file must be a .zip file (or fully compatible).

SUBMISSION REQUIREMENTS

- The full C# solution must be submitted in one zip folder.