CS 3201 – Fall 2020
Assignment 1, 20 pts.
Due: Monday, Aug 31st at 9a.

Objectives

- Become familiar with Visual Studio and C#
- Read data from a text file.
- Create and iterate through a collection of data.
- Create classes that use properties and methods to allow object collaboration.
- Design basic algorithms

Notes

- Please read through all the requirements you begin, so you know what the program is supposed to do when you are done with the assignment.
- Make sure you use ReSharper to verify your code meets the style requirements.
    - Details on how to use ReSharper are available in Moodle.
- **Any program that does not compile will receive a zero (0). Partial credit is not possible for any program that does not compile.**

Overview

This assignment will do some basic analysis of some COVID-19 data. The data will be loaded from a CSV file that is provided in Moodle and that was downloaded from https://covidtracking.com/.

The CSV file is in the following format.

```
date,state,positiveIncrease,negativeIncrease,deathIncrease,hospit
alizedIncrease
```

Data is stored with each record on its own line.

Example of the first few lines of data in a file:

```
date,state,positiveIncrease,negativeIncrease,deathIncrease,hospit
alizedIncrease
20200816,AK,106,4732,0,0
20200816,AL,853,8554,2,0
```

The first line is the header line. The second line is the data of August 16, 2020 for Alaska where there were 106 positive tests, 4,732 negative tests, 0 deaths, and 0 hospitalization on that day. The second line has the same data, but for Alabama.

Requirements

Getting started

1. Download and unzip the Covid19AnalysisA1.zip file as a starting point for this assignment.

2. Unzip the project and open the solution file which will start Visual Studio.

3. On the toolbar for the target platform verify that the x64 architecture is selected.

4. Open the Project properties and under the Build settings. Make sure XML documentation file is checked. It is near the bottom of the Build options.

5. Setup ReSharper to use the class settings file. (We will discuss how to use ReSharper in class on Tuesday, Aug 25th. So this step can be skipped for now.)

6. Build and run the program to make sure the application runs. The GUI should display but there is currently no functionality provided other than updating the text in the output text.

7. Open the Package.appmanifest file and change the Display name: to display your name instead of ....

Required functionality

Add the following required functionality to the application.

1. Add functionality so that you when the user invokes the Load button it will display a FileOpenPicker dialog so that the user can navigate to and load the Covid19Aug16.csv file. A good location for this file is within in the root folder for the project or your Documents library folder.

    a. You will add the code to create and display the FileOpenPicker dialog in the MainPage code behind file MainPage.xaml.cs within the loadFile_Click method.

    b. The FileOpenPicker dialog should be customized as follows:

        i. The following file type filters: .csv and .txt.
        ii. View mode to Thumbnail.
        iii. Suggested starting location to the Documents library.

    c. Note: When displaying the picker and using the await operator it requires the method it is invoked within to be an asynchronous method. To do this you will need to add the async attribute to the method header. If this is done for an event handler method, then it is OK to have an async void method.

2. Add functionality so that after the user loads the data file and the data is loaded, the application will then determine and display the following in the summaryTextBox for **Georgia** data only.

   a. The day of the first positive test in Georgia.
   b. The highest number of positive tests and the date it occurred on.
   c. The highest number of negative tests and the date it occurred on.
   d. The highest number of tests on a day and the date it occurred on.
   e. The highest number of deaths and the date it occurred on.
   f. The highest number of hospitalization and the date it occurred on.
   g. The highest percentage of positive tests and the date it occurred on.
   h. The average number of positive tests per day since the first positive test.
   i. The overall positivity rate of all tests.
   j. The number of days from the first positive test with the number of positive tests greater than 2,500.
   k. The number of days from the first positive test with the number of positive tests less than 1,000.

   The format of the above output is up to you, but it must have labels for all required data that is to be displayed.

3. Add functionality to display a monthly breakdown of the following stats (below) for each month. The output for each month must be preceded by the name of the month. The first month displayed will be the first month with positive test.

   a. The highest number of positive tests and the date it occurred on.
   b. The lowest number of positive tests and the date it occurred on.
   c. The highest number of total tests and the date it occurred on.
   d. The lowest number of total tests and the date it occurred on.
   e. The average number of positive tests per day for the month.
   f. The average number of total tests per day for the month.

   Example output for the monthly breakdown. The day it occurs on must have the st, nd, rd, th ending as appropriate. (Data from the provided file was not used in the example output below.)

```
March
Highest # positive tests: 532 occurred on the 29th.
Lowest # positive tests: 0 occurred on the 23rd.
Highest # total tests: 12,345 occurred on the 12th.
Lowest # total tests: 3,210 occurred on the 2nd.
Average # positive tests: 151.38
Average # total tests: 9,525.00

April
Highest # positive tests: 753 occurred on the 17th.
Lowest # positive tests: 252 occurred on the 1st.
Highest # total tests: 18,712 occurred on the 22nd.
Lowest # total tests: 8,251 occurred on the 3rd.
Average # positive tests: 552.40
Average # total tests: 13,205.42

…
```

4. Formatting:
    a. Numbers should be displayed with commas to denote thousands separators.
    b. All averages should be displayed to two decimal places.

End of requirements

Submission

- Use ReSharper to clean up the code and address all ReSharper warnings.
- Make sure all required documentation is provided and complete.
- Add a notes.txt file to the project and note any missing functionality or bugs.
- Make sure you clean your solution (Build → Clean Solution) and then zip up your project which needs to include the solution file into a ZIP file called *YourName*A1.zip and submit the assignment by the due date.

Grading breakdown

Total 20 points (2 pts – demo file load and parsing; 10 pts. – functionality; 8 pts – implementation)

- *Any program that does not compile will receive a 0. Partial credit is not possible for any program that does not compile.*
- *If a program is only partially complete and a category cannot be accurately assessed, you will not receive full credit for that category.*
- *One point will be deducted from your grade, if your submission does not follow the naming convention.*

Demo file load and parsing due Tuesday, August 25th at 9a - 2 pts.

The submitted application for this will be graded as follows:

- 2 pts. – The application is able to load and display the data in the CSV file, showing that you can parse out the individual fields for a record. How you show this is up to you, but I must be able to determine that you can parse the individual fields of the data. You can do this, by formatting the output for the date, min, and max or having some of the requirements completed that demonstrates you can load and parse the data from the file.
- 1 pts. – Worked on project, but the ability to load the file and demonstrate that can parse a CSV record is not completed. If can display the file contents, but cannot demonstrate that can parse a CSV line you will not earn the full 2 points.
- 0 pts. – Barely started or not demoed. If can only display the FileOpenPicker dialog will earn 0 points.

No rubric will be used for the functionality scores. It will be graded on a continuous scale.

Implementation rubric

If a program is only partially complete and a category cannot be accurately assessed, you will not receive full credit for that category.

Implementation rubric is on the next page.

	Exceptional	Acceptable	Amateur	Unsatisfactory
Readability	2 pts.	1 pt.	NA	0 pts.
Implementation	4 pts.	3 pts.	2 pt.	1 pts.
Documentation	2 pts.	NA	1 pt.	0 pts.

Grading description

	Exceptional	Acceptable	Amateur	Unsatisfactory
Readability	The program is exceptionally well organized, very easy to follow, and there are not any ReSharper warnings.	The code is fairly easy to read and there are a few ReSharper warnings that were not legitimately explained as to why they still exist.		The code is poorly organized and very difficult to read. Many ReSharper warnings remain.
Implementation	The code follows best practices in OO design & program development. (OODD) The code could be reused as a whole or each routine could be reused.	Most of the code follows best practices in OODD development. Most of the code could be reused in other programs.	Some of the code follows best practices in OODD Some parts of the code could be reused in other programs.	The code does not follow best practices in OODD or violates very basic OODD principles. The code is not organized for reusability.
Documentation	The required documentation is well written and clearly explains what the code is accomplishing. No redundant inline commenting.		The required documentation is there, but not complete or is only somewhat useful in understanding the code.	The documentation is poor and very incomplete.