CS 3201 – Fall 2020
Assignment 3, 20 pts.
Due: Monday, Oct. 19th at 9a.

Objectives

- Refactor code to remove code smell
- Add new functionality to existing code
- Create and use properties and methods within a class to allow object collaboration
- Write basic UI code
- Write data to a file.

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.**

Getting started

1. The goal of this assignment is to improve and increase the functionality of the COVID-19 analysis project from Assignment 1 and 2. The starting point of this project is a correct and completed Assignment 2. You will use your submission from Assignment 2 as a starting point for this assignment, if there were bugs and/or code smell issues in your Assignment 2 submission, they must be fixed or full credit cannot be earned for this assignment.

2. The new and improved COVID-19 data analysis project will have improvements in the following areas:

    a. Add missing functionality and remove all bugs from Assignment 1 and 2.

    b. Refactor the code to make use of best practices in class design and clean code that eliminates code duplicity, promotes reuse, and is code that is efficient, readable, and understandable.

    c. Add version control to your project.

    d. Add unit testing to test some of the functionality of the application.

    e. Add functionality to clear out any existing data.

    f. Add functionality so that the user can add new data.

    g. Add functionality so that the user can save the data.

    h. Add functionality so that the user can modify the histogram bucket size.

Requirements

Requirement #1: Complete, debug, and refactor A2

1. Debug and complete all the requirements that have been given so far.

    a. Note: If your project had errors or missing functionality based on the Assignment 1 and 2 requirements, including the use of LINQ, those issues must be addressed in this submission. If these bugs still exist, major points will be deducted from the functionality score.

2. Refactor your code to eliminate all class design and code smell items that have been discussed in class, presented in the book: *Clean Code* and/or noted in any specific feedback.

    Items that need to be addressed include, but are not limited to, the following:

    - Separation of concerns - placing functionality and corresponding state information in appropriate model, view, io, etc. namespaces and classes.
    - Making sure the implementation adheres to best practices, in particular, in regards to SRP and DRY.
    - Eliminate data duplicity within and between collaborating classes.
    - Use appropriate constructs to query a data set, e.g., LINQ.
    - Address readability, e.g.,
        - Appropriate use of enums and/or constants for magic numbers
        - Formatting and ReSharper issues
        - Appropriate number of levels of code within a method
        - Appropriate use of white space
        - Writing self-documenting code
        - Adding required and appropriate documentation

Requirement #2: Github and version control

1. Add version control to your project:

    a. In Visual Studio, add git-based source control to your project. This can be down by selecting Add Solution to Source Control… in the context menu for the solution.

    b. In GitHub, create a **private** git-based repository for this project and link your local repository with the remote (GitHub) repository.

        i. If asks you to pay money to create a private repository, then it means your account is not currently specified as an education account. You can go to https://education.github.com/discount_requests/student_application to request your account be designated as a GitHub Education account.

        ii. To link your repository to GitHub use the Home → Sync section of the Team Explorer and select Publish Git Repo in the Push to Remote Repository section.

    c. Share the GitHub repository with (yoder531) dyoder@westga.edu. This can be done in the Settings → Manage Access section of the repo within GitHub.

    d. Use version control as you develop your project to commit changes as you develop your code. Make sure you commit your changes locally and also push them to the remote repository.

2. In GitHub add Issues for all bugs that need to be addressed and for new functionality that needs to be added.

    a. Use the issue labels to specify whether the issue is a bug, enhancement, etc.

    b. When you have addressed the issues, mark them as resolved.

3. There should be at least eight commits to your repository with appropriate comments for the commit.

    a. The commits should be for functionality that is completed, not just small textual or formatting changes.

4. Once you have completed the assignment tag the completed assignment changeset with the following label Assignment3Submission.

    a. You can create the tag in Visual Studio, but if you do so you will also need to push the tag from Visual Studio.

    b. You can also create the tag in GitHub by creating a release.

Requirement #3: Unit testing

1. Add unit testing for five (5) model methods. (Some details for unit testing are below, but a demo of unit testing in Visual Studio will be done in class on Tuesday, Oct. 13th.)
    a. Select five model methods and add unit testing that provides complete coverage for testing the method.
        i. The methods selected for unit testing need to have some type of logic or perform some type of calculation. Unit tests for a method that is like a getter only do not count toward this requirement.
2. Create a new *MSTest* test project within the solution by right-clicking on the solution in the *Solution Explorer* and adding a new **Unit Test App (Universal Windows)** project to the solution.
    a. Make sure to give the project an appropriate name for the project it is testing.
    b. Make sure to create a unit test app that is the Universal Windows one so that the unit testing will work with a UWP application.
3. Conventions
    a. Unit tests should be in a corresponding namespace as the class being tested.
        i. Then either create an additional sub-namespace that is the name of the class being tested or put the name of the class being tested within the class name for the method that is being tested.
    b. Create one test class for each method that you are testing.
        i. When creating the unit test project, the first test class will be named: UnitTest1 and be found in UnitTest.cs. Either remove this file and class or move it and rename it appropriately.
    c. All the unit tests for a single method - one method for each test case - will be within the same test class. In other words, you will create one test class for each method you are testing.
        i. Use meaningful method names for each test case.
4. At the top of the test class add a comment which lists all the test cases for the method you are testing. The test cases should be written out in the two column format (Input, Expected output) discussed in class.
5. Remember: If you are testing a method that has preconditions, then you will need unit tests for the preconditions and the exceptions that are expected.
6. You should run the tests using the Test Explorer.
7. Make sure to format the unit testing code using ReSharper, but you do not need to worry about using ReSharper warnings on files within the test project.

Requirement #4: Clear existing data

1. Add functionality that will allow the user to clear any existing data that has been loaded or added to the COVID-19 data set.

Requirement #5: Add new COVID-19 data

1. Add functionality to the application so that the user can add a new day of COVID-19 data.
    a. The UI should allow the user to enter all required fields.
    b. The data the user enters should be added to any existing data that is loaded or if there is no COVID-19 data then add to the empty data set.
    c. If the day the user entered already exists, then the user should be prompted whether to keep or replace the existing day.
    d. The UI should ensure the data entered is not negative.

Requirement #6: Save COVID-19 data

1. Add functionality to the application so that the user can save the data out to a CSV file. The format of the file will be the same as currently specified and given below.
2. The user should be able to specify the name of the file to save to using a FileSavePicker.

Requirement #7: Change bin size for histogram breakdown

1. Add functionality so that use can change the bin size for the histogram.
    a. The UI should not allow the user to enter a negative value.
    b. The first bin will go from 0 to the bin size entered.

Requirement #8: Note any issues

1. If there are in bugs or missing functionality, then within GitHub's *Issues* create an issue for each item.

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.
- Tag the submission.
- 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*A3.zip and submit the assignment by the due date. Note: I will clone the project from GitHub, but I want to make sure I have an exported version of the project.

Grading breakdown - 20 points (12 pts. – requirements; 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.*
- *No rubric will be used for the requirements/functionality grade. 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.

	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.