Object-Oriented-Programming Assignment #1

Liam Whitehead (ID: 25698752), March 2022

Code reviews I performed -

I provided code reviews for two people. The reason for this was to produce higher quality, longer and more versed reviews instead of making many small reviews. I reviewed code from Samuel Jonathan Orman-Chan (ID: 25659005) and Luke-A-C-Roberts (ID: 25722923). The latter of the two had a partially unfinished project which allowed me to suggest features that could improve functionality and their grade. The former, however, was more oriented towards better variable naming schemes and small quality improvements as they had finished and improved theirs upon the review of others before me. When reviewing someone's code, the easiest method for me was to clone the repository straight to my GitHub desktop workspace then open it in my preferred IDE (Visual Studio 2022) to pick apart and analyse. I would run the program first for any typical errors and erroneous inputs, then with that in mind go through the code in the intended work flow. This allowed me to understand their method and suggest improvements. Documentation was easy to point out as I went along.

Code reviews I received -

I received 4 reviews coming from LukasSob (ID: 25728866), Nathan Edwards (ID: 25725278), Samuel Jonathan Orman-Chan (ID: 25659005) and Stevie Goodman (ID: 25175912). Links will be provided at the end of this document. Some reviews held more formative comments than others, but overall they majorly improved the quality of my program and opened my eyes to new techniques that helped to streamline and create a better program. None of the reviews could be seen as unhelpful as at the bare minimum they pointed out flaws and problems with running the program. Once I rectified each reviewers problem, then there would be less problems for other reviewers to encounter. Specific modification/changes I made from reviews include:

Analyse.cs

- Nice use of Regex.
- Nit: Regex could be shorter for both vowels and consonants, and by converting the case within the scope of a loop for instance, the redundant use of Upper & Lower Case in the regex strings could be avoided. For instance:
  `// Define Regex Filter and create Regex object for filter string vowels = @"([aeiou]){1}"; Regex rxVowels = new Regex(vowels); // Create a MatchCollection object for the matches of the Regex filter from the input MatchCollection matchVowels = rxVowels.Matches(input.ToLower()); values[1] = matchVowels.Count;`
- Consonants could also be made as `^[aeiou]{1}`.
- No count for individual letters/characters.

(From Samuel Jonathan Orman-Chan, ID: 25659005)

The regex simplification advice here drastically changed how I implemented my filters. It reduced all of my filters by at most half. Also, I later did implement the individual characters count. I started off with just letters, but added other topics of analysis such as numbers & whitespace & punctuation.

Report.cs

- Nice and pleasant output.
- Nit: As most terminals work in a monospace (typewriter) font, you could use whitespace characters to format the output in neat columns.
- `Console.WriteLine` on line 17 does not need to have any arguments passed to create a newline. In-fact you could just add the `\n` escape sequence to the start of the following line.

(From Samuel Jonathan Orman-Chan, ID: 25659005)

From this review of my Report.cs class, I was inspired to completely change my output technique by organising them into pre-allocated areas that have been programmed to have neat columns. I also started using \n to start a new line and it has made my program a small bit neater.

Input.cs

- Add using System.IO else '!Directory.Exists(fileName)' on line 32 cannot be used
- System.IO.FileNotFoundException when the file input does not exist
- Good and clear commenting

(From Nathan Edwards, ID: 25725278)

This section of review pointed out an extreme flaw in my code, the ! Character was flagging an error and I had not specified using System.IO yet. It was an easy fix but allowed reviews later on to focus on other areas.

-- File Text Choice
The search by text file works great, but it could output the text so the user knows what piece of text has been analyzed in the program.

(From Luke-A-C-Roberts, ID: 25722923)

Though a small quality of life tip, I implemented it and It made the output information more understandable.

Improvements

- You can move a lot of the functionality from `Program.cs` into `Input.cs`. This would improve abstraction, encapsulation and your grade.
- Add exceptional handling for invalid paths.

(From Stevie Goodman, ID: 25175912)

After this section of feedback, I modified my code by taking almost a third of the code in Program.cs and instead used it in Input.cs. This helped me keep the Program.cs class for the important areas such as method calls to other classes.

Links -

- My review of Samuel Jonathan Orman-Chan (ID: 25659005) : https://github.com/SJO-C/CMP1903M/issues/8
- My review of Luke-A-C-Roberts (ID: 25722923): https://github.com/Luke-A-C-Roberts/CMP1903M-Assessment-1-Repository/issues/4
- Review from LukasSob (ID: 25728866): https://github.com/LWhitehead15/ObjectOrientedProgramming-Assessment-1/issues/7
- Review from Nathan Edwards (ID: 25725278): https://github.com/LWhitehead15/ObjectOrientedProgramming-Assessment-1/issues/6
- Review from Samuel Jonathan Orman-Chan (ID: 25659005) : https://github.com/LWhitehead15/ObjectOrientedProgramming-Assessment-1/issues/5
- Review from Stevie Goodman (ID: 25175912): https://github.com/LWhitehead15/ObjectOrientedProgramming-Assessment-1/issues/4
- My GitHub Repository: https://github.com/LWhitehead15/ObjectOrientedProgramming-Assessment-1