Soft Skills Assignment - How to Write Effective Documentation

Deadline: Wednesday, September 16 2020, 09:00 AM

GitHub Classroom Repository

Introduction

For this assignment you will research a topic from the list of approved topics below. You will also review the materials provided on How to Write Effective Documentation. Your task is to create a Google Document where you will write your own technical documentation to explain the topic that you have researched to beginner programmers.

Requirements

- Google Doc
    - Document must be publicly accessible
- Github Repository
    - Must contain working examples of the topic you have researched
    - Code must be commented
        - Comments must be spell-checked
        - Comments must be grammatically correct
    - Code must comply with the official C# Coding Conventions.
    - Repository must be publicly accessible
    - README.md must contain a link to your Google Doc
    - README.md must also have the introduction of your documentation
    - README.md must have a citation summary
- Presentation to be delivered on deadline day
    - Aim for 3-5m (not under 2m or over 10m)
    - Explain use cases, inputs, outputs, potential exceptions, etc

How to Write Effective Documentation

This slide deck will give you guidance on writing software documentation.

Topics (Pick ONE)

Simpler or Previously Discussed Topics:

- Research and demonstrate a practical use of a read only or derived property in an object (properties with only a getter, arrow notation).
- Research and demonstrate how to validate a value coming into the setter of a fully implemented property (properties with a code block for get and set).
- Research and demonstrate how to format the value coming out of the getter of a fully implemented property (properties with a code block for get and set).
- Research and implement a namespace-level enumeration that is used in multiple independent classes (not derived from one another).
- Research and document the behaviour of object initializer lists and what happens when some properties are left out of the list.
- Research the format of a lambda expression (using an anonymous function) and what the various segments of the expression represent.
- Research when explicit casting is necessary, when it is invalid (compiler error), and when it throws an exception.
- Research and implement a custom user-defined exception derived from the Exception class.

Intermediate Topics:

- Research and demonstrate a practical use of Operator Overloads for a custom type in C#.
- Research and demonstrate a custom IComparable Interface for comparing two instances of a custom type in C# that you would like to make sortable.
- Research and demonstrate use cases that compare and contrast Implicit Casting vs. Explicit Casting in C#.
- Research and demonstrate how to split a single class into multiple .cs files using Partial Classes in C#.
- Research and demonstrate how to write xUnit tests using IEnumerables for Theories (you will also need to use Yield).
- Research and demonstrate Lambda expression syntax being used with LINQ aggregates (Sum, Count, Min, Max) on a Collection.
- Research and demonstrate how to use Enumerable.Aggregate() function to sort a list based on a custom rule (like sorting a list of names alphabetically based on the second letter of their name).
- Research and demonstrate how to use Enumerable.Join() function to join two Collections that have a shared key into a single Collection. (Used as Demonstration)
- Research and demonstrate how to use Enumerable.GroupBy() to create grouped sublists based on a Collection in C#.
- Research and demonstrate how to use the Enumerable.OfType() method to replace a typeof()-item.GetType() based Enumerable.Where() call.
- Research and demonstrate how to use the Assert.SubSet() and Assert.SuperSet() methods of XUnit in order to verify that a series of items are present in a Collection.

Challenge Topics:

- Research and demonstrate a topic that you choose
    - Topic must be approved by TECHCareers staff.
    - Topic must be more advanced than the “Simpler or Previously Discussed Topics” listed above.
    - Topic must include C#.

Rubric

Requirement	**Points**
Google Document ● Tutorial is in a publicly accessible Google Doc ● Tutorial is free of spelling and grammar errors ● Tutorial uses simple language ● Tutorial uses descriptive titles and subtitles ● Tutorial has at least one annotated screenshot that is not a screenshot of code ● Tutorial has at least one code example in code that someone can copy and paste ○ The code example looks different than the surrounding text ● Tutorial uses figures, lists, quotes, charts, headings, etc to help break up large blocks of text to avoid reader exhaustion	8
Github Repository ● Must contain working examples of the topic you have researched. ● Code must be commented ● Comments must be spell-checked ● Comments must be grammatically correct ● Code must comply with the official C# Coding Conventions ● Repository must be publicly accessible ● README.md must contain a link to your Google Doc ● README.md must also have the introduction of your documentation ● README.md must have a citation summary	9
Challenge	**Points**
Publish your work for the world to see. ● Find a platform such as Medium, Dev.to, or reddit.com/r/learnprogramming and create a professional account ● Post your work following all the rules of etiquette provided by the platform ● Provide a link to this posting on your README.md	3
Summary	**Points**
Requirements:	17
Challenges:	3
Total:	20

Hints & Tips

We have provided a sample Google Doc here for you to use as an example. You do not have to follow the template or formatting provided in this document. This is just a starting point. Feel free to look at other tutorials for inspiration.

Sample GitHub Repository

Sample Solution Google Doc

Reminders

Ensure you are tracking your time and that your timesheet is in the appropriate folder for viewing and marking.

Ensure you are committing frequently. It is advised to commit once per successful feature implementation at a minimum.

Ensure you are pushing your repository to the GitHub Classroom repository, and not a personal, private repository.

Ensure your repository has a readme with at a minimum your name, the name of the project, the project's purpose and a link to your Trello board.

Ensure you are using Trello appropriately to keep track of outstanding features, and that it is linked in the README.md file.

Ensure you are using the #class channel to request clarifications on assignment specifications.

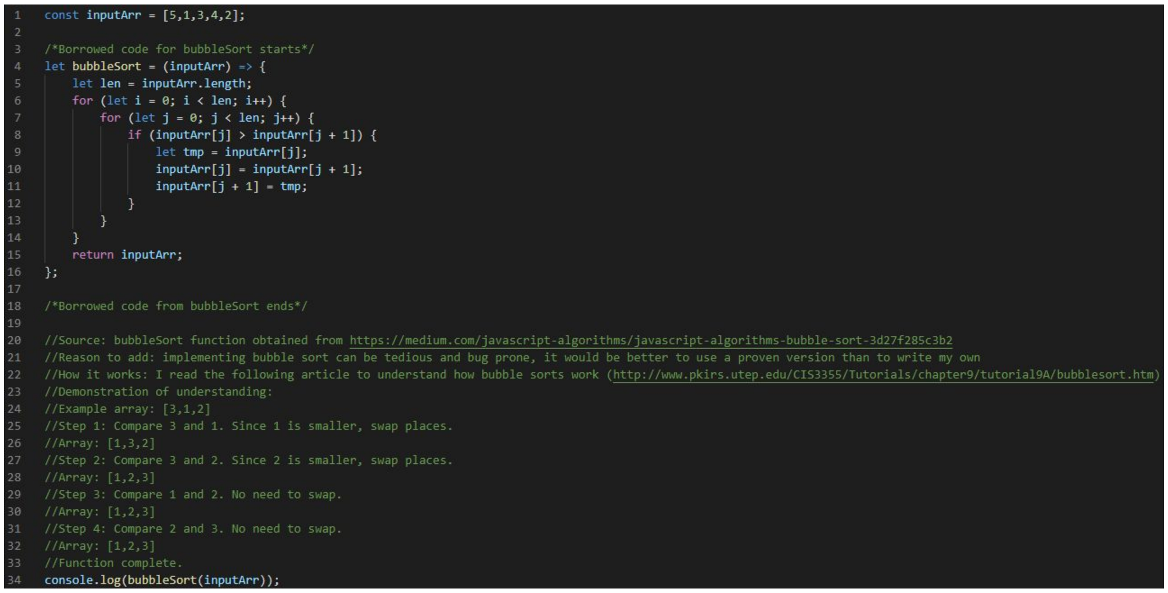
Ensure you are using the #homework-help channel in slack to request for assistance from instructional staff if needed, and that you include (at a minimum) a specific description of the problem and a list of what you have tried.

Feel free to reach out on #peer-support if the #homework-help queue is lengthy. If someone helps you out, give them a shoutout using our handy-dandy form!

Citation Guide for Borrowed Code

Whenever you borrow code, the following information must be included:

- Comments to indicate both where the borrowed code begins and ends.
- A source linking to where you found the code (URL, book, example, etc.).
- Your reason for adding the code to your assignment or project instead of writing it out yourself.
- Explain to us how the code is supposed to work, include links to documentation and articles you read to help you understand.
- A small demonstration to prove you understand how the code works.



```javascript
const inputArr = [5,1,3,4,2];

/*Borrowed code for bubbleSort starts*/
let bubbleSort = (inputArr) => {
    let len = inputArr.length;
    for (let i = 0; i < len; i++) {
        for (let j = 0; j < len; j++) {
            if (inputArr[j] > inputArr[j + 1]) {
                let tmp = inputArr[j];
                inputArr[j] = inputArr[j + 1];
                inputArr[j + 1] = tmp;
            }
        }
    }
    return inputArr;
};

/*Borrowed code from bubbleSort ends*/

//Source: bubbleSort function obtained from https://medium.com/javascript-algorithms/javascript-algorithms-bubble-sort-3d27f285c3b2
//Reason to add: implementing bubble sort can be tedious and bug prone, it would be better to use a proven version than to write my own
//How it works: I read the following article to understand how bubble sorts work (http://www.pkirs.utep.edu/CIS3355/Tutorials/chapter9/tutorial9A/bubblesort.htm)
//Demonstration of understanding:
//Example array: [3,1,2]
//Step 1: Compare 3 and 1. Since 1 is smaller, swap places.
//Array: [1,3,2]
//Step 2: Compare 3 and 2. Since 2 is smaller, swap places.
//Array: [1,2,3]
//Step 3: Compare 1 and 2. No need to swap.
//Array: [1,2,3]
//Step 4: Compare 2 and 3. No need to swap.
//Array: [1,2,3]
//Function complete.
console.log(bubbleSort(inputArr));
```