Assignment02-text

Explain how you apply the self-documenting principle and encapsulation principle in the above code for Questions 1-3

Assignment02-Q4:

a. For the first question, I utilized the self-documentation principle by creating comments only when necessary to allow a reader to quickly parse the code and understand its use. In addition to that, I used a class name of `Converter` and method name of `OctalConversion` to tie the method class name and create a simple execution statement in the main function, being: `converter.OctalConversion(x)`.

b. In the 2nd question, I commented on the necessary pivot points in the code base (moving from each encapsulated code block to the next). Specifically using variable names such as: `numbers3_5, numbers3, numbers5` to signify the divisible numbers by modulo #. Furthermore, encapsulation was used to create a private method named `Modulator` to only return an array of strings divided by category. This was a clear division of purpose, relative to the `Main()` which only received input and produced output.

c. Lastly, I utilized self-documentation by defining a clear function `Tree(int n)` which produced the required right-angled tree. Following that I purposefully used `y` as the first variable in the nested for-loop to represent the height of the tree, and `x` to represent the horizontal length of that line. Encapsulation allowed me to create the Tree method and separate its purpose of producing the tree, to the `main()` 's function of producing the output to the user.

Assignment02-Q5:

1. Benefits of Self-Documentation & Encapsulation
    a. The benefit of using the principle in the first question is to allow for easy code readability and maintain the longevity of the code base. Given the naming conventions and easy readability, it can be used to add further

Converters to the same file. The benefit of encapsulation for this question is to keep all converter types to themselves. In this case, we only had to use Octal as an example but adding more conversion types would mean we can maintain a method to each of those processes and maintain a good understanding of all its components.

b. Self-documentation allowed me to clearly identify which batch of strings I was working with when it came to `numbers3_5, numbers3, numbers5`. Furthermore, encapsulation clearly divided the method's purpose of performing all the logic and the main's method of receiving the input and producing the output.

c. If I had not utilized self-documentation and encapsulation I could have produced various iterations of that tree by accident, not producing code with a clean flow and purposeful execution. This would have been most apparent in the nested for loops, that without careful consideration of the boolean logic can add unwanted values.

2. Drawbacks

   Across the board, the drawbacks are a more convoluted approach to creating larger and larger systems. Self-documentation improves the longevity and ease of understanding of all code bases. Encapsulation ensures that all methods, classes, or code blocks are serving a single purpose without breaching multiple different purposes.