Assignment02-Q4:

For Question 1, I apply the self-documenting principle by ensuring variable, class, and method names are descriptive, such as the getOctal() method, which returns the octal. I apply the encapsulation principle by ensuring that variables are only accessible within their methods. The Octal() constructor method has access to all the data it needs, such as the input integer and all the other variables, and it doesn't share the data or the process until the return statement. Additionally, only methods within the class can access the Octal() method.

For Question 2, I apply the self-documenting principle by having descriptive method names as well as comments to explain small parts of the code. I apply the encapsulation principle by ensuring that each method is only responsible for one small part, and each small part only has access to the data and process it needs to do its job. For example, divideByThree() only has access to the input integer and the numbers that are divisible by three. It doesn't need to know which numbers are divisible by 5 or both.

For Question 3, I apply the self-documenting principle by ensuring the counter variable is clearly named, as well as adding necessary comments to improve understandability. The variable, class, and method names are very clear. I apply the encapsulation principle by ensuring that Triangle() is the only method to have access to the process of printing the triangle. I could have done this in the main method, but creating a new method and class for this process allows me to better apply the principles of OOP.

Assignment02-Q5:

The benefit of applying the self-documenting principle is that your code is easy to read and follow, which means anyone else working on your code can easily add to it or make changes. The readability of code is incredibly important when working with large teams of software engineers. The drawback of not applying this principle is that your code will be hard to follow. Any collaborators, including yourself, will not have any idea what is going on. When you look back at the code, you will be confused and will have to take a lot of time to understand it again.

The benefit of applying the encapsulation principle is that each method will have access to only the information it needs, which allows you to better apply the divide and conquer principle. It allows you to know exactly which methods have access to specific data. The drawback of not applying this principle is that it can be hard to modify or add to if changes need to be made. It can be confusing to determine which methods have access to what data and therefore, any changes made in the future can potentially cause unintended consequences.