Assignment02-Q4: The encapsulation principle is applied in Questions 1-3 by all methods being private so that access to them is restricted to only within their respective class, hiding their internal workings. The classes also do not utilize any public variables so that they cannot be accessed by outside objects.

I applied the self-documenting principle by using descriptive variable names, like “decimalNum”, “octalNum”, “remainder”, “isNegative”, etc. in Q1, “div_3”, “div_5”, and “div_both” in Q2 for the lists of integers divisible by 3, 5 and both respectively, and “row” and “col” in Q3 in the nested for loops. In Q2 and Q3 I use an integer called “n” because that is what is specified by the problems. I also use descriptive names for methods in Q1 and Q2, which directly state their respective functions: “convertDecimalToOctal” for converting an input decimal to an output octal number, and “listToOutput” for taking a list and returning the formatted string to match the output specified by the problem. The class names are also descriptive and reflect their function, being “OctalConverter”, “FactorsApp”, and “RightTriangle”. Beyond naming, the code structure in each is simple and logical, minimizing the need for extra comments.

Assignment02-Q5:

1. The benefit of applying the self-documenting principle in questions 1-3 is that other developers can easily understand what the programs are doing because of the descriptive names and uncomplicated code, making it easy to implement, edit, or add to the systems. It’s also beneficial to me to apply this principle in these problems so that I can build a habit of writing clean, self-documenting code.
   The benefit of applying the encapsulation principle is that the methods and variables in the classes can only be accessed from within the class, improving security, and by keeping the program in one class there is no repetition or unnecessary complexity with communication between multiple objects. Encapsulation also improves maintainability and modularity by keeping the methods and data isolated and easily reusable in the future.
2. The drawback of not applying the self-documenting principle, ie. not using descriptive names and possibly having overcomplicated, difficult to understand structures, is that it may be difficult for other developers, or even the original developer, to understand the program and make changes, implement it in a larger system, etc.. Or, there may be unnecessary comments that only waste time or are ignored.
   Not applying the encapsulation principle might look like having multiple classes to answer each of the problems, which would be unnecessarily complicated and messy, and would involve some variables and/or methods being public, allowing them to be easily accessed by any outside command. Overall, the security would be weaker with information not hidden, and the programs may be unnecessarily complicated, making it more difficult for developers to work on implementing or editing the code as they have to work with more classes and more code overall. Additionally, lack of encapsulation can lead to harder-to-maintain code due to dependencies between classes