Assignment02-Q4:

The self-documenting principle was applied by using descriptive variables names and adding comments to explain what is going on in the code. For example, the toPrint strings clearly indicate that it will be printed, and octalNumber states that it is an octal number. The method names also describe what is being done, such as displayTriangle or getDivisibleNumbers. The encapsulation principle is applied by using separate classes and methods for different purposes, and using private variable that can only be accessed by these methods in scope. The main method in App.java serves to pass the initial parameters into the other classes, which then use private variables that are only edited within the class, like in DivisibleNums with the num variable only being edited within the getDivisibleNumbers method. This way, the internal state is kept protected.

Assignment02-Q5:

The benefit of the self-documenting principle is it makes it easier for other people to understand it, even those without programming knowledge. This also makes it easier to implement future changes as it is clear what is already going on in the code, making it essential for bigger operations. The benefit of the encapsulation principle, it reduces the possibility of unintended modification of variables which reduces errors in the program. It also makes it easier to debug as each method can be tested independently, while making it easier to scale with more methods without affecting the entire program.

The drawbacks of not applying the self-documenting principle are creating an overreliance on comments, which leads to big unnecessary walls of text that make it hard for others. The same goes with having undescriptive variable names, as it makes it harder for others (or yourself in the future) to understand what is going on in the code, making it harder to test. The drawbacks of not applying the encapsulation principle are the exact opposite of the benefits, it makes it harder to extend the code base as variables and methods are too codependent, which also increases the possibility of bugs. With public variables, any method or user can edit them which exposes your codebase to external threats and introduces problems with modularity. Not encapsulating also leads to unnecessary extra code to deal with the exposed variables.