**Assignment02-Q4:**

Within my code to answer questions 1-3, I showcased the self-documenting and encapsulation principles effectively. To begin, I used the self-documenting principle by using descriptive and straightforward method names such as “getUserDecimal()”, “convertToOctal()”, “trianglePattern()”, as well as variable names like “userNum”, “decimal”, and “octal” to make the code readable and comprehensible for others. This minimizes the need to provide comments explaining the function of each line of code. I implemented the encapsulation principle by organizing my work into separate methods such as handling user input, converting from decimal to octal, and pattern printing. This allowed me to call these methods into the main method of the program, in which I was able to execute them in only a couple of lines. By integrating the concept of modularity, where methods are focusing on a single task, it further helps myself and others to understand the code better as it’s a lot more organized and easy to follow.

**Assignment02-Q5:**

There are many benefits of applying the principles of self-documenting and encapsulation, especially within my code for questions 1-3. Specifically, in regards to the self-documenting principle, it promotes improved code readability as using descriptive variable and method names, the code is much easier to understand without needing a lot of comments to explain what the code is doing. This essentially helps others to quickly grasp the functionality of the code and as the code explains itself, there is less of a need to create external documentation, allowing for time to be saved and reducing inconsistent commenting. Furthermore, regarding the encapsulation principle, it has a great benefit as it allows for functionality separation into various methods, with each method performing a single, specific task. In addition, encapsulation enables developers to debug code easier and it lets them reuse methods across the program over many times, which reduces redundancy and development time.

Some drawbacks of not applying the self-documenting principle include, difficulty in code readability and increased efforts in maintaining the code. Specifically, if the code uses vague, misleading variable and method names, it becomes very hard for someone to understand the code they are reading, especially if they are unfamiliar with the given code. Furthermore, the lack of clear variable and method naming results in the developer to include more comments or documentation to explain their logic behind the code, which is very inefficient and can be inconsistent over time as they make updates to their program in the long-run. The drawback of not applying the encapsulation principle includes poor modular code, difficulty in debugging and testing, and reduced reusability. Particularly, if the code is not divided into modular methods, the code becomes very difficult to manage and changes to the code will inevitably affect other parts of the program if the logic of the code isn’t divided properly. Moreover, it becomes harder to debug, test, and reuse individual functionalities of the code if they aren’t separated effectively as finding the errors will be harder to identify, fix, and maintain.