Assignment2 - Q4

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

I used the self-documenting principle by commenting on my code, ensuring all variable or method names are self-explanatory and short. I ensured to explain complex arithmetic or parts of my code that may not make sense to others.

I used the encapsulation principle by running any arithmetic or changes to the input outside the main class, and into a static method. In this method, all operations to the input are made in the method, so it allows for readability of what operations are done on the input. I prevented direct access to the modification of the input.

Assignment2 - Q5

Explain the following:
1. What's the benefit of applying the self-documenting principle and encapsulation principle in the above code for Questions 1-3?
2. What's the drawback if you do not apply those principles?

Self documenting and the encapsulation principle are essential to ensure the code is readable and others can understand and track what modifications are made to an object. Encapsulation allows for the people in a project to ensure all operations to an object are in one manageable spot. Encapsulation is what makes debugging an easier process. Self documenting can allow for easier optimization if others can see what your code is supposed to do.

If I did not use these principles, all the operations to the object are done in the main class, which creates a clumped and less readable bunch of code. It is harder to track what operations are done to an object as it could be anywhere in the program. Without the self documenting principle, I could name variables or methods anything I like, including names that do not relate or make sense at all. It creates a program that is harder to read and debug. If there is ever an issue, the team will require a lot more time to track what part of the program is causing a semantic error.