# **Open/Closed Principle**

## **Advantages:**

* Extensibility: can extend functionality without modifying already existing code, reducing bugs

e.g., Through implementing the problem using the IShape Interface, the calculation of the area is easily done implementing the CalculateArea() method within the Triangle Class, as opposed to changing the GeometricShapes Class which can result in errors that are harder to debug.

* Maintainability:

e.g., It is more intuitive to look for the area of the triangle within the Triangle Class than some other class that might implement the functionality, which makes the code written using OCP easier to maintain and refactor.

* The interface IShape provides flexibility, allowing for changes within the implementation of CalculateArea, without changing IShape, as well as modularity and separation of code, making the project easier to understand and maintain.

## **Disadvantages:**

* Complexity: using OCP requires more code to implement which would lead to an increased complexity

e.g., For the project without OCP, there are only 3 classes for the shapes and one that handles the calculation of the area, while for the one containing OCP, there is an additional interface. This is not a problem for this small project, but the complexity could be seen in a bigger project, like the Vending Machine project during which the increased number of interfaces created made it a bit more confusion to look through.

* Abstraction: While adding more abstraction into the code, this might make it harder to understand and debug.

e.g., This problem was encountered more during implementing functionalities in the Vending Machine project, as stated above.