- You can improve the design model to create a good encapsulation .
- Inappropriate name of classes (input , output ....).
- Triangle is a specialization of ConvexPolygon
- Isosceles, scalene, and equilateral are Triangles.
- No strategy design

Anas "Mohammad Ramadan" Ahmad AlSobeh , Sep 21 at 1:52am

Improving Design to create good encapsulation:

I don't know how this design could have been encapsulated better. I identified three main areas where design might change in the future. Each one of those design decisions was encapsulated in a class, one for Input methods, one for Output methods, and one for AreaFinding operations. It's true that there are no access control specifiers to enforce people from breaking my encapsulation, but that is a Python design decision.

Inappropriate name of classes:

This is a subjective opinion. Input and Output handle all of the Input and Output for the program, why should they be called anything different?

Triangle is a specialization of ConvexPolygon:

If this is a comment about the UML diagram, then I accept that Triangle was not listed. In the code, however, I've listed Triangle as a specialization of ConvexPolygon on line 33 of AreaFinder.py.

Isosceles, scalene, and equilateral are Triangles:

Same as above, with code examples on lines 38 and 46 of AreaFinder.py. A scalene triangle is exactly identical to a "triangle", so I saw no need to have two objects for them.

No strategy design:

Both Input and Output use the strategy pattern. Python defines a function as first-class data, so the functions are defined below the class definition. The functions are contained by composition in the Input and Output class in the `strategy` variable, where they are later called. The function of the Input and Output classes changes dramatically based on the strategy selected.

- No meaningful attributes and operations
Anas "Mohammad Ramadan" Ahmad AlSobeh , Sep 21 at 1:53am

No meaningful attributes and operations:

I assume this is about the UML diagram. I have no complaints with this.

So, according to the rubric, I feel like I deserve:

A clear and concise model of Shapes using UML Class Diagrams - 15 pts.

A working implement, with good encapsulation, abstractions, inheritance, and aggregation, and appropriate use of the strategy pattern - 45 pts.

Meaningful, executable unit test cases - 30 pts.