5.1P: In Person Check-in 2 – Answer Sheet

1. What was the most challenging aspect of the drawing tasks? Why?

From my perspective, the most challenging aspect of drawing tasks is drawing rectangles and lines. While drawing borders for both rectangles and circles is relatively straightforward, drawing line for the task 4.1 requires more efforts than I thought. Finllay, after spending whole day, I realized that drawing line is not hard although it makes me struggle for a little bit..

2. Review your answer to question 3 from check-in 1. Did you use any of the strategies you identified? How did they go?

Personally, when facing with challenging tasks, I frequent consider potential solutions to solve the problems. After that, I utilize all acadmic materials including e-books, textbooks, and useful Youtube channels not only to figure out answer but also gather knowledge for solving complex problems and self-learning. Consequently, I reuse code in task 2.3P for 3.3P task. However, I had changed code a little bit in task 4.1P. As a result, all tasks are completed based on instructions and my own creativity.

3. What are some strategies for success you can start or continue using for the remainder of the semester?

In my point of view, for the remainder of the semester, I will spend more time for assignments and feedback from tutor, allowing me to learn valuable mistakes from previous weekly exercises. Morevoer, I will access Youtube channels and read academic materials for extending academic knowledge. Finally, I will enhance solf skills include time management and priority to get better grades.

4. Summarize how you utilized Encapsulation, Abstraction, Inheritance, and Polymorphism in your submitted tasks.

Based on the submitted tasks 2.3P, 3.4P, and 4.1P, Encapsulatiom, Abstraction, Inheritance, and Polyorphism are utilized:

+ Encapsulation: Burdles data (***_data*** and ***_radius***) with the methods that operate on it within a class. Therfore, private fields (***_x, _y, _width, _height, …)*** keep the data internal and inaccessible from outside. Public properties and methods provide controlled, safe access to modify or retrieve the data, allowing to protect the object's integrity and internal state

+ Abstraction: Defines a common interface for objects, specifying what they can do (for example ***Draw()*** and ***IsAt()*** methods) without telling how. Therefore, the ***Shape*** class is abstract, serving as the blueprint. Moreover, it hides complex implementation specifies, offering interaction at a higher and more generic level.

+ Polymorphism: means " many forms" and enables different types of objects ***(MyCircle, MyRectangle, MyLine)*** to be treated as a comon ***Shape*** type. When a method like ***Draw()*** or ***IsAt()*** method is called, the specific version beloging to its actual type is executed, allowing the ***Drawing*** class to manage and interact with various kinds of shapes.

+ Inheritance: allowing new classes ***(MyCircle, MyRectangle, Myline)*** to reuse properties and behaviors from an existing base class (***Shape***).