## ( / 45) Assignment 2: OOP

**( / 35 ) Task 1:**

You are provided with a starter code with the parent class Shape and a main test code. Implement the following classes to **complete this program and have a running main**. You are expected to use coding best practices, encapsulation that we have seen in class, and reuse code where possible. Example enforcing a member method be constant when applicable.

Your submission must include a .PDF file with the written solution and all your code files for the Task implementation.

**Rectangle, Circle, and EquilateralTriangle classes: (12 points + 3 points for coding style and best practices)**

* Override inherited member methods. ( 3 \* 3 points )
* Constructor implementation (3 \* 1 point)

*Notes: The displayProperties should display all class information (e.g., For rectangle: width, height, area, perimeter)*

**GraphicEditor class:** Implement the following methods **(15 points + 5 points for coding style and best practices)**

Attributes:

* A std::vector<Shape\*> called shapes to store shape pointers.

Methods:

* addShape(Shape\* shape): Adds the shape to the shapes vector. (2 points)
* displayAllShapes(): Loops through each shape in shapes and calls displays properties for each. (2 points)
* Shape\* largestShape(): Finds and returns a pointer to the shape with the largest area. (3 points)
* void displayLargestShape(): displays the properties of the shape with the largest area. If no shapes are available, display an appropriate message. (2 points)
* double totalArea() Calculates and returns the sum of the areas of all shapes in shapes. (2 points)
* void clearShapes(): Deletes all shapes in shapes and clears the vector. (2 points)

hint: use the .clear() method to remove all elements from the vector container.

* Destructor: Ensures all dynamically allocated shapes are deleted to prevent memory leaks. (2 points)

The output of main is as follow: A black screen with white text

Description automatically generated

**( / 3) Task 2:** Use a UML class diagram to describe the relationship between the different classes.

**( / 2) Task 3:** In our particular implementation example what is the association (composition vs aggregation) that best describes the relationship between GraphicEditor and Shape. Explain.

Composition:

In this implementation, GraphicEditor manages a collection of Shape pointers. While aggregation would allow shared ownership, GraphicEditor takes full responsibility for creating and deleting Shape objects. This strong ownership aligns with composition, as Shape objects do not exist independently of GraphicEditor, they are created, managed, and deleted by it.

**( / 5) Task 4:** Mark each statement as either **True** or **False.**

1. T\_\_\_ - The operator 🡪 combines de-referencing and membership when used with a pointer.
2. \_F\_\_- A C++ array will not allow you to supply index values that are out of range of the array.
3. \_F\_\_- Only one class can be implemented at a time in a \*.cpp file.
4. \_F\_\_- The SDD cannot change once it has been written.
5. \_T\_\_- A UML Class diagram presents a static view of classes in a system, their members, and the inheritance structure.
6. \_F\_\_- Variables cannot be declared outside a function or class.
7. T\_\_\_- The copy semantics provided by the default C++ Copy Constructor is member wise copy
8. \_F\_\_- we can rely on the default copy constructor if we are using raw pointers as class attributes.
9. \_\_T\_- A sprint should always result in a demonstrable deliverable (i.e. working increment).
10. \_T\_\_- Default arguments should be assigned in a function prototype, or declaration, only.