An Object Oriented Drawing and Painting Application

Objectives:

Upon completion of this assignment you will be able to:

- Abstract and model entities from reality in the form of classes
- Design an object oriented model for geometric shapes
- Draw a UML class diagram that represents your model
- Apply the OOP concepts of inheritance and polymorphism to your design
- Create an advanced GUI with 2D Graphics capabilities

Part 1: Geometric Shapes Data Model

Description

Geometric shapes belong to different groups (ex: Elliptical Shapes, Polygons, Sectors…etc…). Members of these different groups are related to each other in the sense that they share common properties. In order to be able to implement an efficient and object oriented drawing application. It is essential to design a model that takes these relations into consideration.

Tasks:

- Design an object-oriented model that covers the following geometric shapes: Line Segment, Circle, Triangle, Rectangle and Square.
- Identify **at least 4** required **Design patterns** that can help to facilitate

the implementation of the Application

- **Draw a UML** (**Class and use case) diagrams** that represents your model, showing all the classes' attributes and methods.
- Apply the 4 components of **OOP** to your design.
- You should apply **SOLID** principles in the program.

Part 2: Drawing and Painting Application

Drawing and painting applications are very popular and have a huge user base. They generally offer a big number of features that includes but is not limited to: Drawing, Coloring, and Resizing. They allow the user to undo or redo any instructions so as to make the application more usable.

Tasks:

- Implement your design from part 1 in an OOP language.
- Design and implement a GUI that allows the following functionalities for the user on all the shapes defined in part 2: Draw, Color, Resize, and Move.
- Implement your application such that it would allow the user to undo or redo any action performed.
- The cursor should be used to select the location of a shape while drawing it, or moving it to another location, for more accurate control on the shape parameters (ex: size), dialog boxes could be used, or you're free to implement it in a more user friendly way of your choice.

Deliveries & Notes:

- ✓ You should write the program using **java language**.
- ✓ You should work in **group of three**.
- ✓ Your code should be **clean**, **readable** and **commented**.
- ✓ You should deliver a **report**, contains description of your implementation you have implemented.
- ✓ You should deliver a report that includes the relationships between entities you are modeling **(UML)**, their attributes, how encapsulation has been followed, user guide, as well as snapshots for your GUI.
- ✓ **Late submission** is accepted and is graded out of **50%.**
- ✓ Delivering a copy will be awfully penalized for both parties, so delivering nothing is so much better than delivering a copy.

Tutorials:

- https://docs.oracle.com/javase/tutorial/uiswing/components/dialog.html
- http://docs.oracle.com/javase/tutorial/uiswing/learn/settingup.html
- https://www.youtube.com/watch?v=LFr06ZKIpSM
- https://docs.oracle.com/javafx/2/get_started/jfxpub-get_started.htm
- https://netbeans.org/kb/docs/java/gui-functionality.html#Exercise_1
- http://www3.ntu.edu.sg/home/ehchua/programming/java/j4a_gui.html

Good Luck