Part 1:

This software project is a Shape Factory that will randomly generate 6 shapes that could be a circle rectangle or square. The goal of this project is to design and practice object oriented design principles to create clean and professional code.

Some problems that I may encounter when developing this software project is getting used to the JPanel and JFrame interface. Some problems with the JPanel may be putting the button and printing the shapes on the same interface while updating the 2d graphics after interacting with the sort and load buttons.

In terms of the classes, I will be using multiple OOD principles to allow the classes to interact in a way that it does not conflict each other. The Square class will inherit attributes and methods from the rectangle class which will also inherit some abstract methods and attributes. The circle class will also inherit attributes and abstract methods. Abstraction of getter methods of color and area will allow attribute to be secure and private while allowing other methods in other classes such as ShapesFactory to get those attributes. There is not much cases of encapsulation but in the getter method of the area there will be a small calculation for the area depending on the type of shape.

A design method that I believe will be beneficial for this software project will be the prototyping model. I believe this is the best for the project since not many changes will be required for the project and since I am not too familiar with the JFrame interface I will be practicing with different prototypes before fully developing all the code. Therefore, I will start with planning and analyzing the requirements of the project followed by creating UML diagrams that will help me visualize the classes and their behaviors and interactions with the other classes. Next will be building the Jframe along with the buttons and shapes that should be in the panel. I will continue prototyping until I can complete working buttons along with the shapes in the panel with its corresponding positions. Then I will go into the development of all the code starting with the shapes classes then going into the logic of loading and sorting in the ShapesFactory class and BubbleSort class. Finally testing will be used to ensure proper interactions with the users.

Part 2:

I believe that the structed diagram and class diagram is the best model. I described each shape and its relationship with the shapes class and the other classes such as BubbleSort and ShapesFactory. In each class I showed each attribute and methods along with symbols that show if it is public, private and protected.

I believe that another model type that could be used is the behavioural model to show how the methods and classes interact with each other. It can show parameters and which classes will call out methods. The most relevant methods for the shapes classes is the getArea class which calculates the area. For the ShapeFactory class the most important methods is the constructor method ShapeFactory since it creates the random shapes.

Graphical user interface, application, Teams

Description automatically generated

Part 3:

The sorting algorithm that I will be using is Bubble Sort. This sorting algorithm has a runtime of O(n^2) at its worst. It is one of the simplest sorting algorithms. It compares and swaps adjacent indexes until the list is fully sorted. In the best case scenario, the sorting algorithm would have to compare and swap only one set of items but the worst case scenario is if the list is sorted in the opposite way meaning that there would have to be multiple passes/repeated iterations. Graphical user interface, application, Word

Description automatically generatedGraphical user interface, application

Description automatically generatedGraphical user interface, application

Description automatically generated

For the video see attached video.

Conclusion:

The stuff that went well in this software project is the implementation of the sorting algorithms and classes of shapes. I think because I practiced a lot I was able to implement it really easily.

Something that did not go well was the implementation of the JFrame components such as the Jbutton and the JPanel. This might have been because this interface is very new to me. It became so bad that I was not able to put the button and shapes in the same panel.

Something that I have learned in this project is how to plan and design proper code. Before, I would code and fix without any direction. Since this project was simple enough to code I believe that I was able to implement software design principles properly.

My top 3 suggestions to complete this project is: make the UML diagram first, then play around with the Jframe components and finally code the shape classes before the ShapeFactory class.