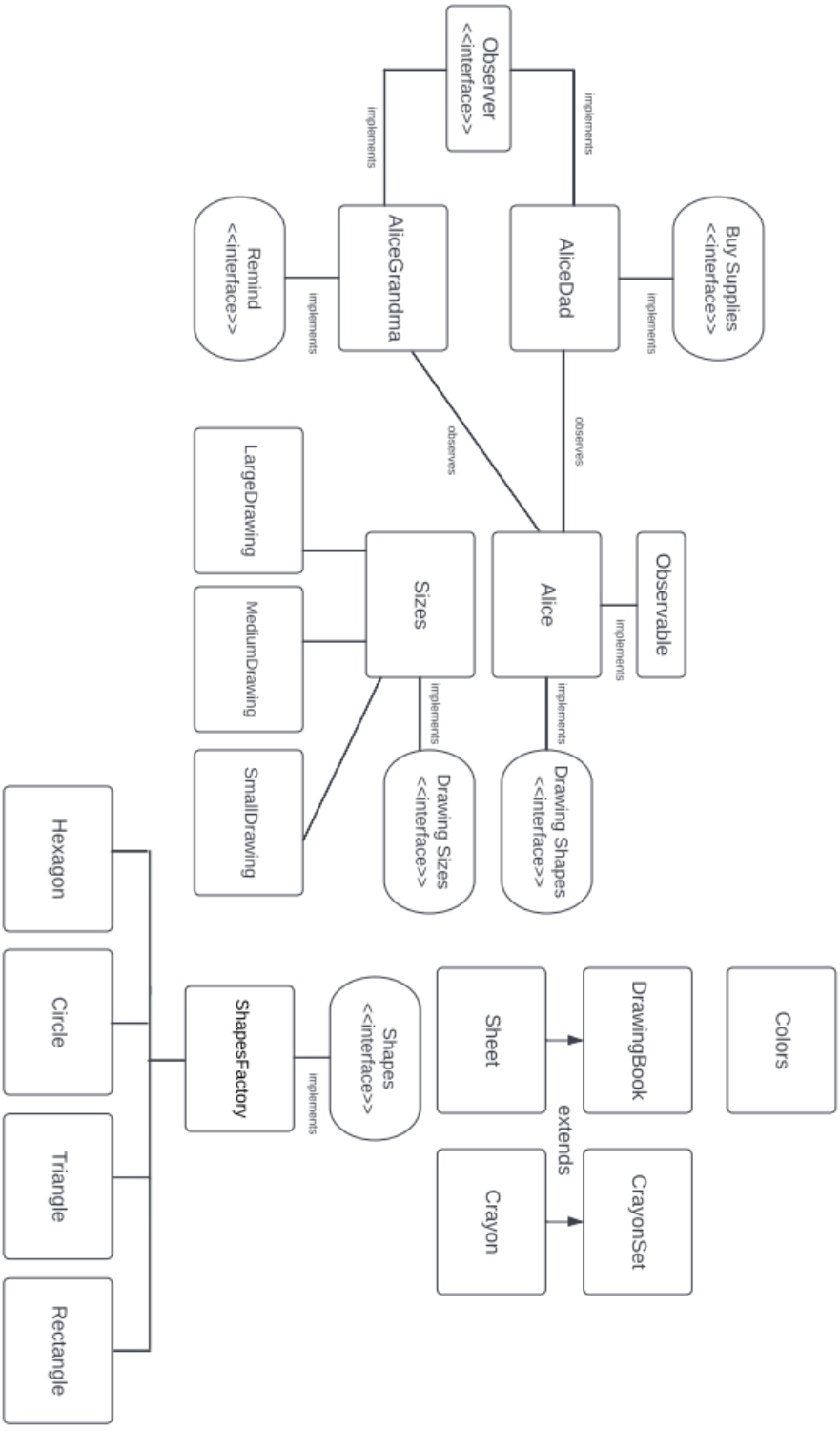
Sami Cemek

OOD Assignment 1

1. Alice, AliceGrandma, AliceDad, Circle, Triangle, Rectangle, Hexagon, Sizes, Shapes, ShapesFactory, SmallDrawing, MediumDrawing, LargeDrawing, DrawingBook, Sheet, CrayonSet, Crayon, and Colors are good candidates for classes. I kept in mind the S.O.L.I.D principles when I'm designing the class structure of this project. Therefore I have quite a lot of classes).
2. DrawingShapes (will be implemented by Alice), DrawingSizes (will be implemented by Sizes), BuySupplies(will be implemented by AliceDad), Remind (will be implemented by
3. AliceGrandma), Shapes (Circle, Triangle, Rectangle, Hexagon) will implement Sizes. It's good practice to use interfaces because they force you to implement methods. In this story, characters have some specific tasks they need to do. Using interfaces to ensure the characters are doing their job (implementing the methods) is a good idea so we don't forget to implement those methods.
4. Alice will compose DrawingBook and CrayonSet. The DrawingBook will compose Sheet. Sheet will compose DrawingShapes. CrayonSet will compose Crayon. Crayon and DrawingShape will compose Colors.
5. Decorator is a structural design pattern that lets you attach new behaviors to objects by placing these objects inside special wrapper objects that contain the behaviors. Just like we did in Week 2 discussion, we can create shapes with different sizes and colors using the Decorator pattern. For example, to make colorful shapes, we can have a class ColoredShape that implements Shape. We can add color to the constructor of the ColoredShape as an argument. So if we draw a shape it can be colorful since we passed color as an argument. We can use the exact same design pattern for size too. (by passing size as an argument of the constructor.)
6. The Observer pattern is used when we need to maintain a "consistent" view of our data across many different data objects, some of which are likely to change values frequently. In this case, we want to let AliceDad and AliceGrandma know under different conditions. AliceGrandma will keep an eye on how many shapes Alice draws in one sitting (max 3). AliceDad will keep an eye on the status of the crayons and the drawing book. So therefore Alice will be Observable, AliceGrandma will be Observer and AliceDad will be Observer.
7. Factory pattern is used to create objects without exposing the instantiation logic to the client and refer to the newly created object through a common interface. As an example, Alice needs to draw shapes, but instead of creating them directly using the new operator, she asks the factory object (ShapesFactory) for a new product, providing information about the type of object it needs. The factory instantiates a new concrete product and then returns to Alice the newly created product(casted to abstract product class). The client uses the products as abstract products without being aware of their concrete implementation.

8.


Observer
<<interface>>
implements
AliceDad
implements
Buy Supplies
<<interface>>
implements
AliceGrandma
implements
Remind
<<interface>>
implements
observes
observes
Alice
implements
Observable
implements
Sizes
implements
Drawing Sizes
<<interface>>
implements
Drawing Shapes
<<interface>>
LargeDrawing
MediumDrawing
SmallDrawing
Colors
DrawingBook
Sheet
extends
CrayonSet
Crayon
ShapesFactory
implements
Shapes
<<interface>>
Hexagon
Circle
Triangle
Rectangle