Report HW3 – Composite, Factories, and Flyweight Design Pattern

In this assignment I implemented a basic shapes library having Point, Lines, Circle, Rectangle and Triangle to understand the practical implementation of Composite, Flyweight and Abstract Factory pattern. AllShapesEnum is a java Enum used for type safety.

Composite Pattern:

1. **Base Component** – Base component is the interface for all objects in the composition, client program uses base component to work with the objects in the composition. It is an interface (shapes) with some methods common to all the objects (Common class).
2. **Leaf** – Defines the behavior for the elements in the composition. It is the building block for the composition and implements base component. It doesn't have references to other Components.
3. **Composite** – It consists of leaf elements and implements the operations in base component.

Abstract Factory Pattern:

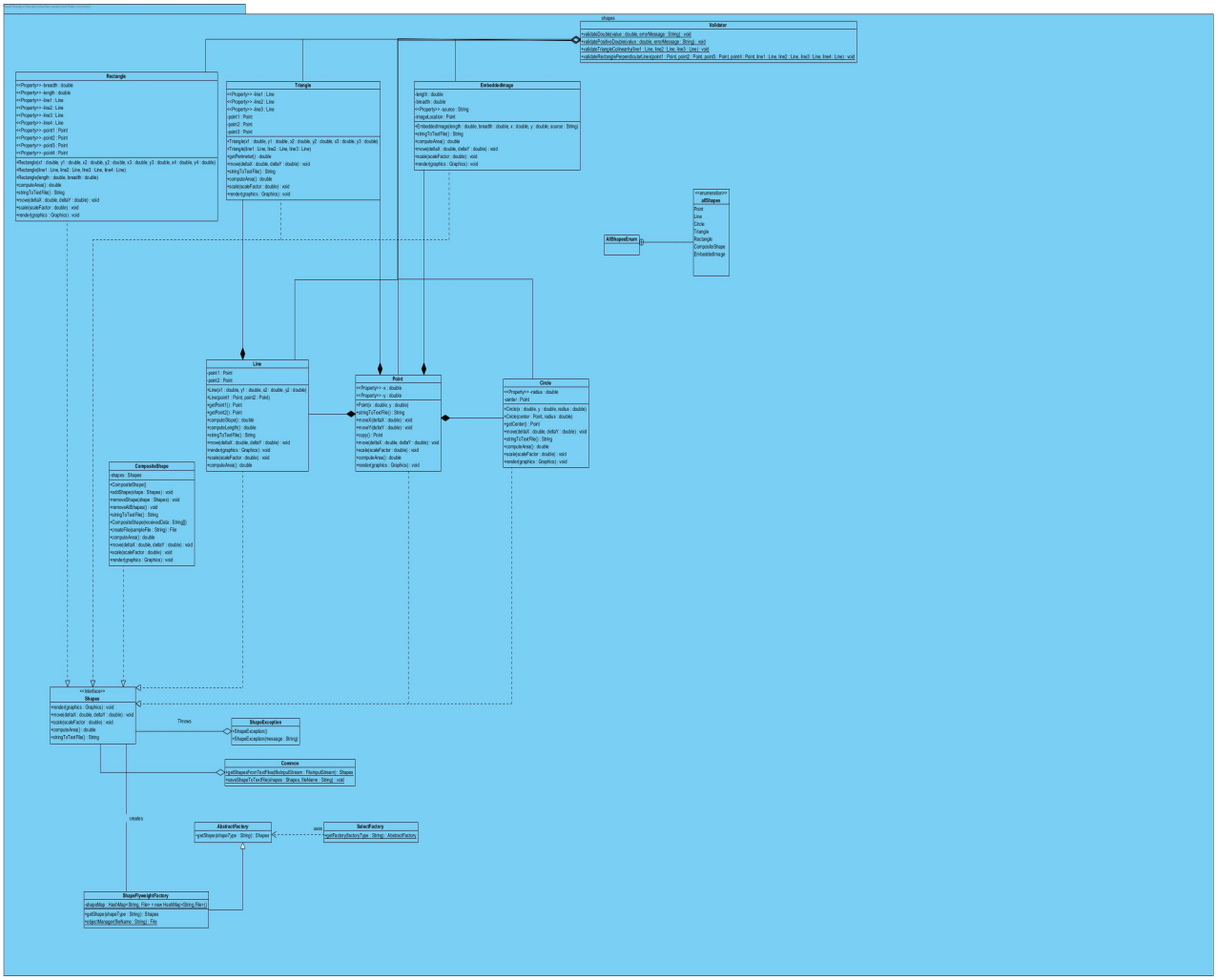
Abstract Factory patterns work around a super-factory which creates other factories. This factory is also called as factory of factories. In Abstract Factory pattern an interface is responsible for creating a factory of related objects without explicitly specifying their classes. Each generated factory can give the objects as per the Factory pattern. The SelectFactory class selects the factories at the moment there is just one factory but can be extended for other derived shapes.

Flyweight Factory Pattern:

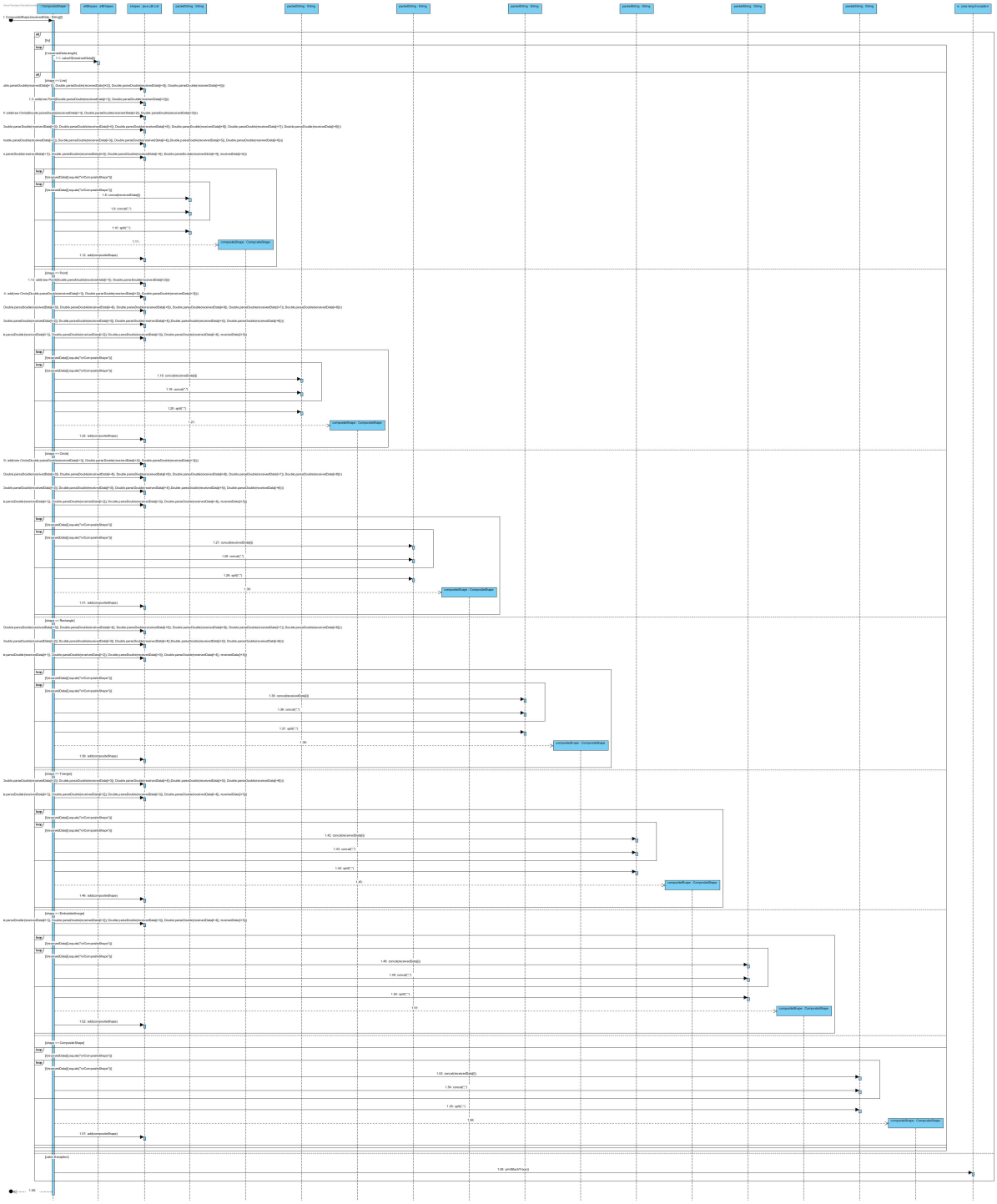
Flyweight design pattern is a **Structural design pattern.** So we will have an interface Shape and its concrete implementations as Line, Point, Circle, Rectangle etc. The Size is an intrinsic property. The flyweight factory will be used by client programs to instantiate the Object, so we need to keep a map of Objects in the factory that should not be accessible by client application. In the Flyweight class a hashmap is implemented to keep a map of objects in this assignment.

Whenever client program makes a call to get an instance of Object, it should be returned from the HashMap, if not found then create a new Object and put in the Map and then return it. We need to make sure that all the intrinsic properties are considered while creating the Object.

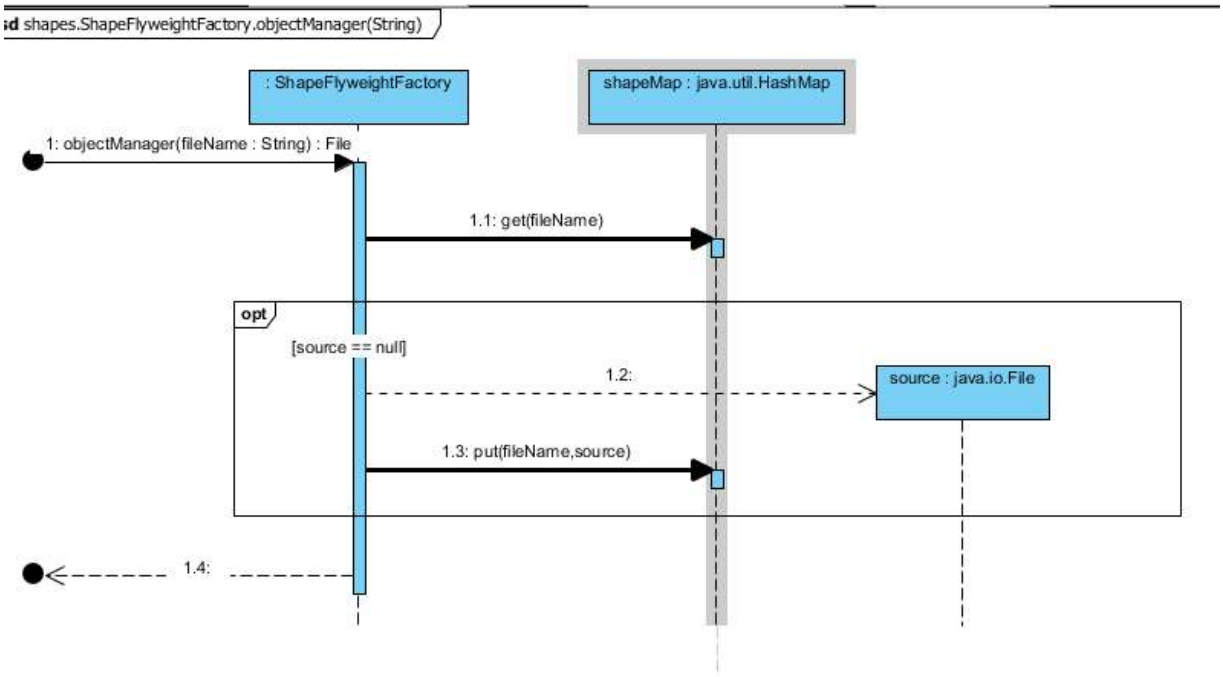
Class Diagram: *(also zipped to the project folder)*



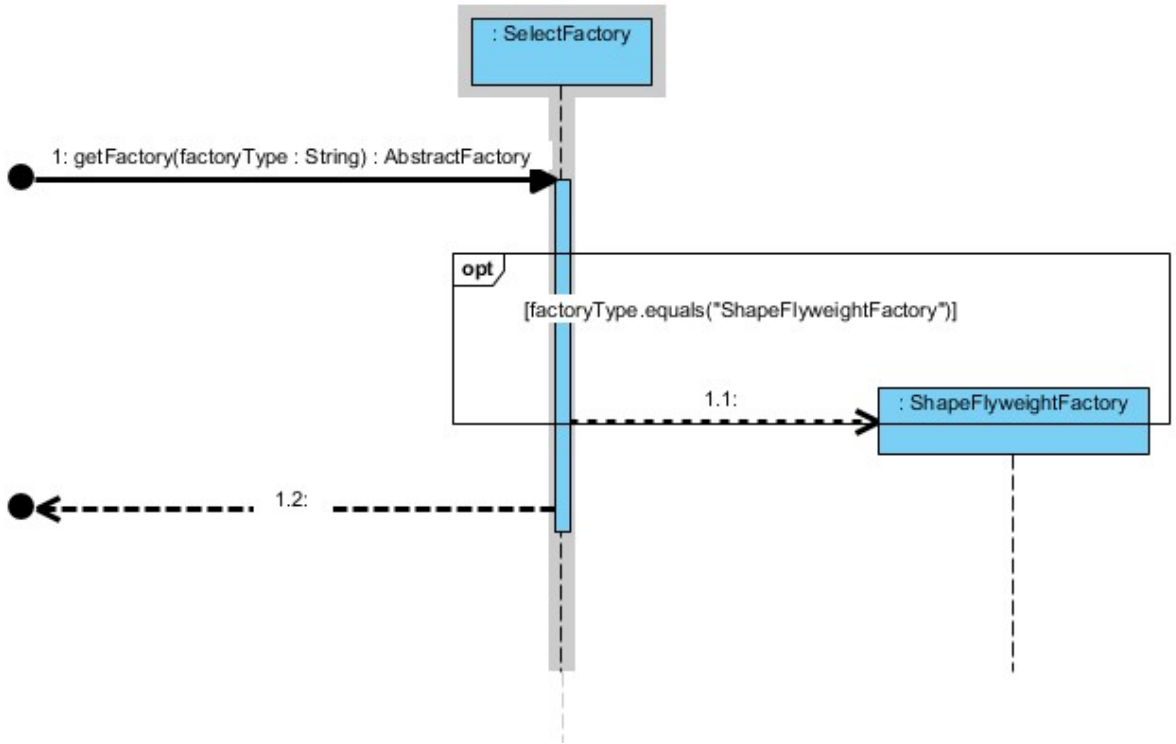
Interaction Diagram: Composite Shapes



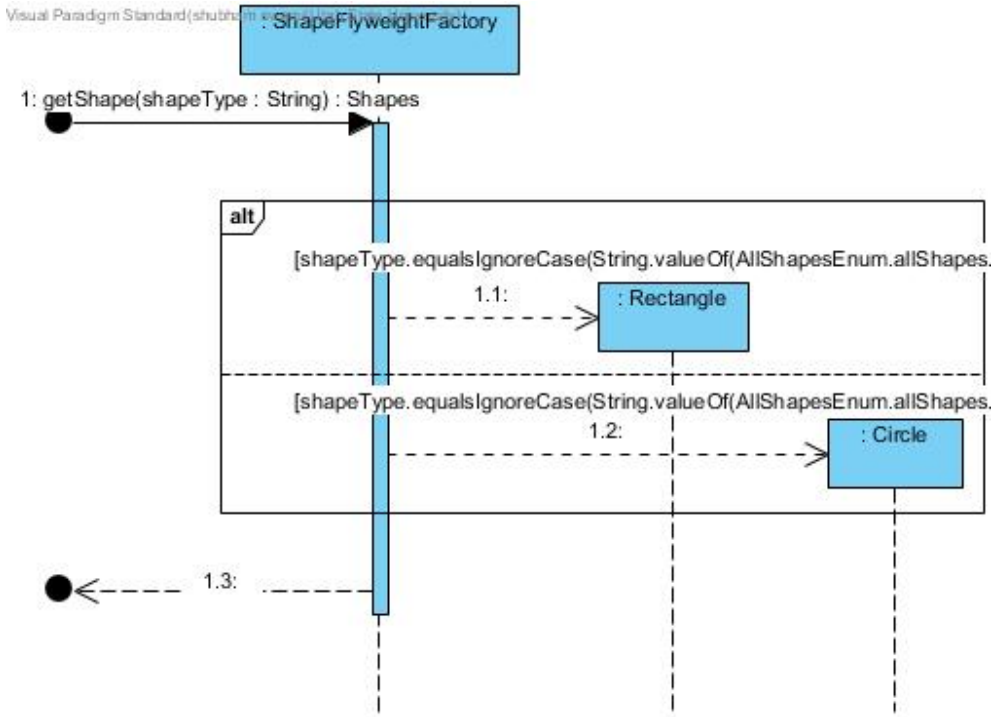
Interaction Diagram : Flyweight



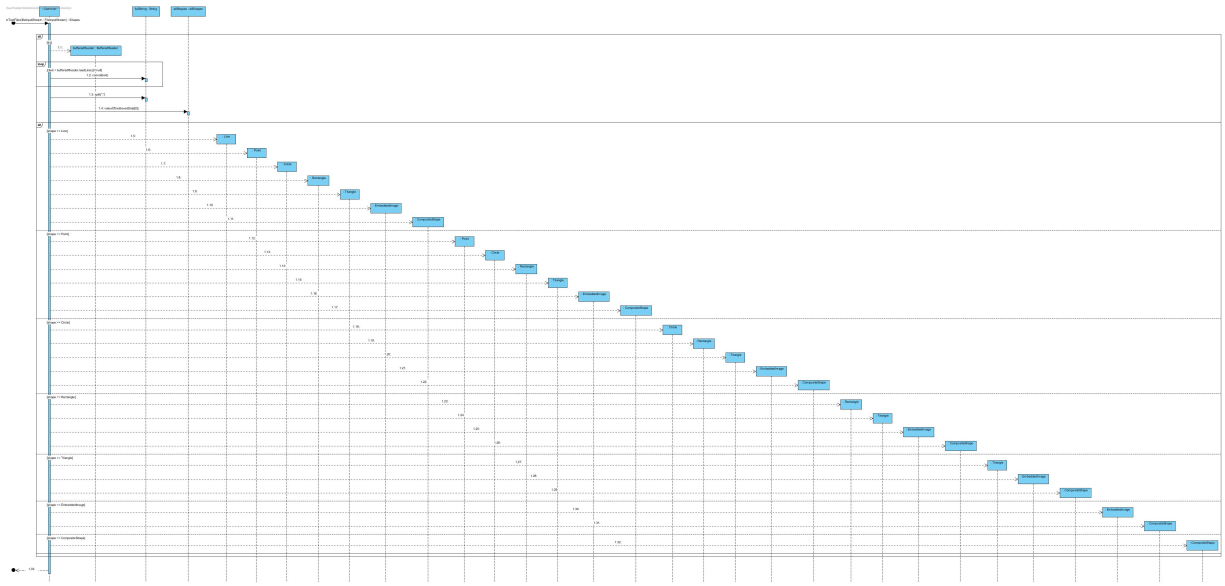
Interaction Diagram: Shape Flyweight Factory



Interaction Diagram: get Shape



Interaction Diagram: Get Shape from text Common



Interaction Diagram : Save Shape to Text common