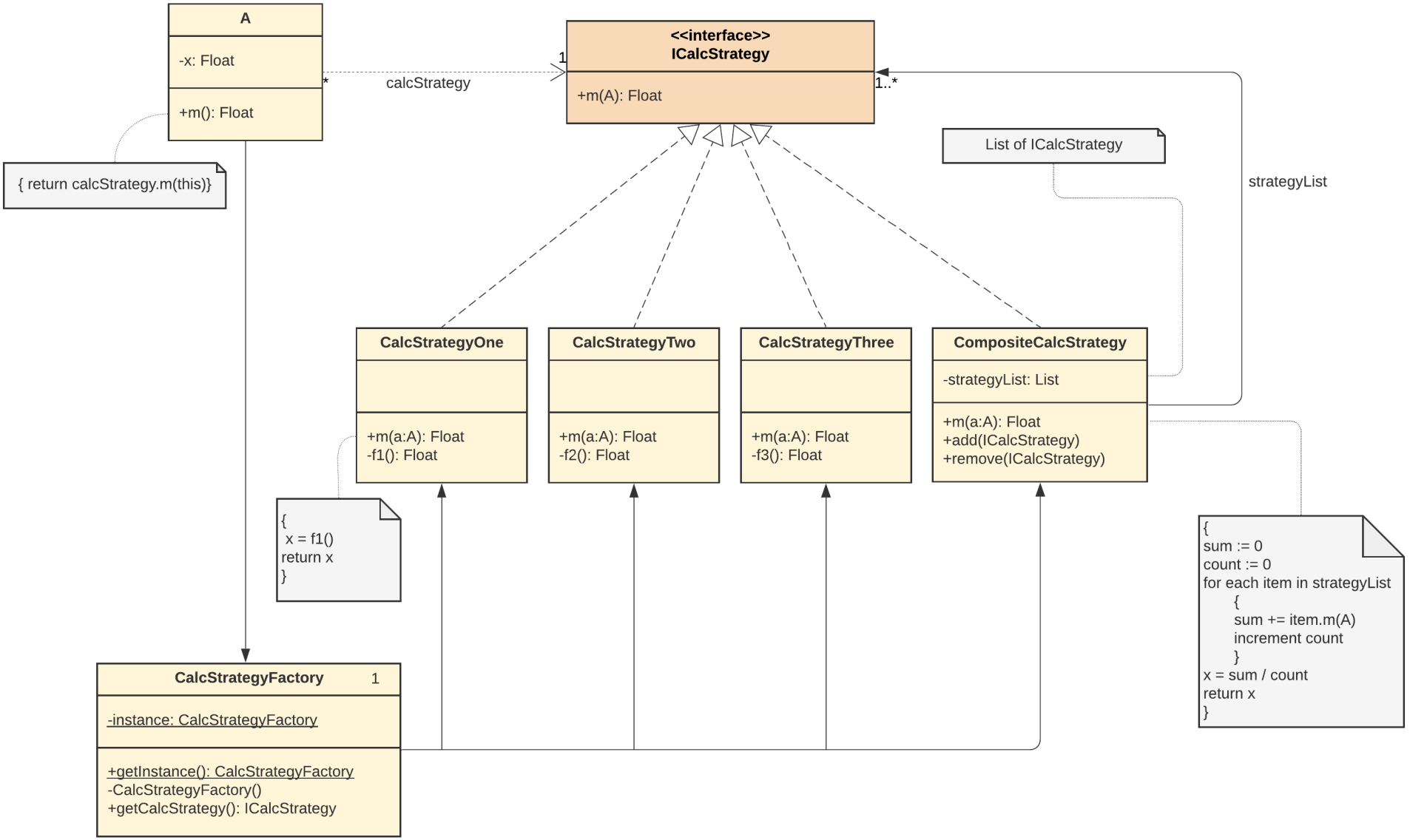
Object Oriented Modeling and Design 5th Assignment

1. UML Class Diagram



2. Design Patterns

2.1. Factory

"Factory Pattern" is a creational pattern. A commonly asked question during system design is "Who is responsible for creating this object?". Factory pattern is responsible for creating related object groups. In our case, *CalcStrategyFactory* is a factory class that is responsible for creating strategy objects.

2.2.Singleton

"Singleton Pattern" is another creational pattern guaranteeing that only a single object will be created from the class. Singleton object is created and returned by a static method of the class. Constructor of the class is private in order to prevent additional object creations. In our scenario, we only need one *CalcStrategyFactory* object in order to create the strategies. Therefore, singleton pattern is used.

2.3.Strategy

"Strategy Pattern" is a behavioral pattern that is used when there are multiple behaviors a program can have. In our case, there are multiple functions (*f1, f2, f3* and more can be added) that can be used to calculate the value of x. Each calculation strategy is encapsulated by a separate class (*CalcStrategyOne* etc.) and these classes are reached from a common interface (*ICalcStrategy*).

2.4.Composite

In our problem, it is stated that multiple functions can be used in order to calculate x. Strategy pattern provides using only one of the functions so that we need another pattern that enables using a combination of multiple strategies. "Composite Pattern" is a structural pattern that eases working with collections. An additional composite class including a list of strategies (*strategyList*) is created. This list can contain both atomic and composite objects. Differently, *m* method of this composite class iterates over the list and returns list average by calling *m* method for each item.

3. UML Sequence Diagram

Implementation note: It is stated that initially, m method uses only function f1 to calculate x. For this reason, the sequence diagram below doesn't show creation phase of csone, which is an instance of *CalcStrategyOne* class that implements f1.