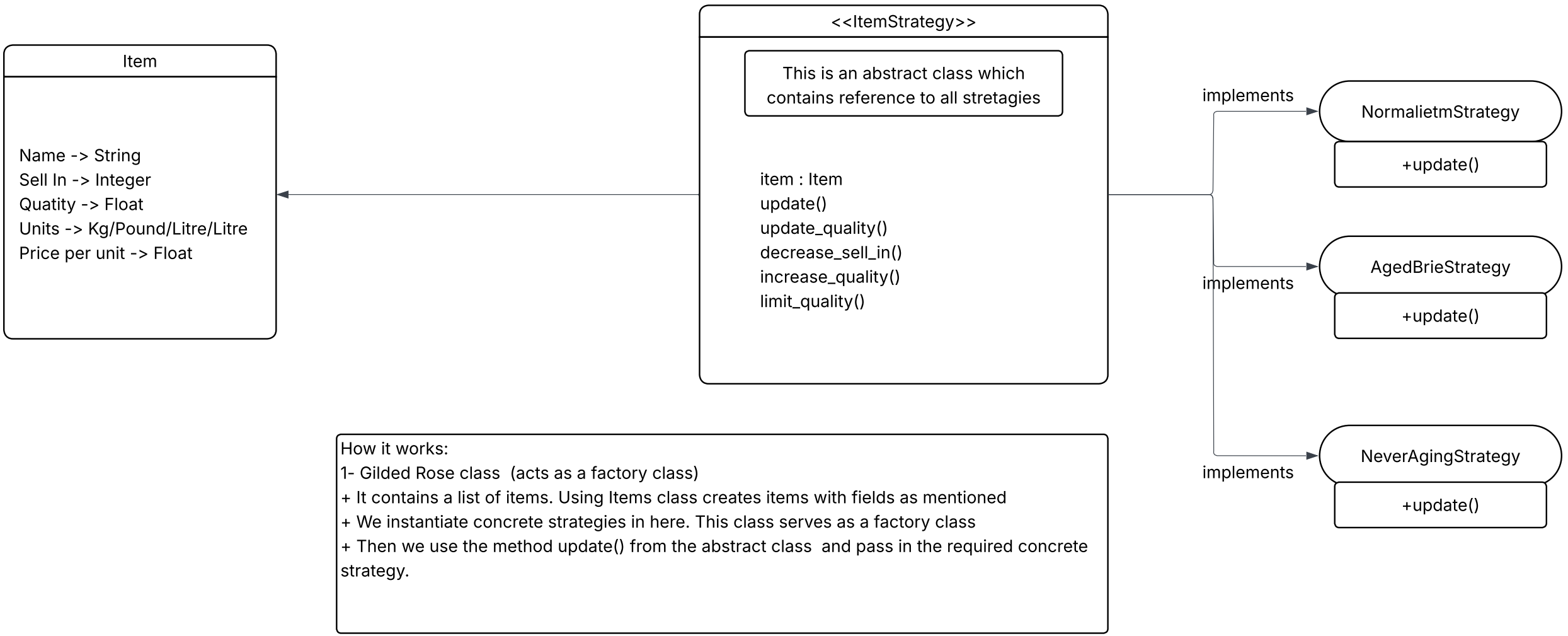
Issues with existing codebase:

+ It was tighly coupled. The methods has brute forced item names and conditions that are not flexible at all.
+ The code needs refactoring if we try to add in a new type of item..
+ We have to manually write in the conditions if we are going to further implement a way to add in a type of product.. (strategy pattern solves this and makes it easy to extend multiple ways of updating based on item..)

Description and Rationale:

I have tried to impelement the strategy pattern because we are performing the same operation but with different ways depending on the object/item.



How it works:
1- Gilded Rose class (acts as a factory class)
+ It contains a list of items. Using Items class creates items with fields as mentioned
+ We instantiate concrete strategies in here. This class serves as a factory class
+ Then we use the method update() from the abstract class and pass in the required concrete strategy.