# Observer Pattern

## Description of the design problem

There is too much coupling between StoreItem, MachineryController, DispenseController Objects.

It means that whenever there is any operation to change the quantity of StoreItem (Coin or Drink), for example if there is a need to add a use case to clear all drinks from the store for maintainer, the MachineryController and DispenseController may need to be called as a part of the operation.

Also if there is a need to add a use case to sync the storage data with an online data Centre, all operations that will change item’s quantity must be modified.

To sum up, it violates the Open for extension, Close for modification Principle.

## Candidate design patterns considered

Since the problem is caused by tight coupling between class, and it is a behavioral issue, the candidate patterns are

* Observer
* Mediator

## Motivation to choose a pattern that would solve the problem including support for new requirement s or changes to existing problems

In this case Observer Pattern is suitable.

The nature of problem is how to assure the consistency of StoreItem’s quantity between multiple Objects without making the classes tightly coupled.

By introducing Observer Pattern, it reduce the coupling because no matter what operation that change the StoreItem’s quantity, the operation itself does not need to worry about make the related objects to be consistent, hence it does not need to know what are these related objects.

Furthermore, it becomes easier to extend to support new requirement for both side (Subject and Observer) due to both depend on abstraction. Changes to each side will not impact another.

## Structure of the pattern (you should map the participants to your applications classes/objects)

## Collaborations among the participants (specific to your application objects)

## Implementation decision that you have taken

1. Mapping subjects to their observers.

It is a valid issue in this case since there are two type of subjects with 11 instances in total, comparing with only 2 or 3 observers. Let every object of Coin to maintain an Observer list which contains the same contents. Same as Drinks. On the other hand, instead of maintain the subject-observer mapping in a hash map