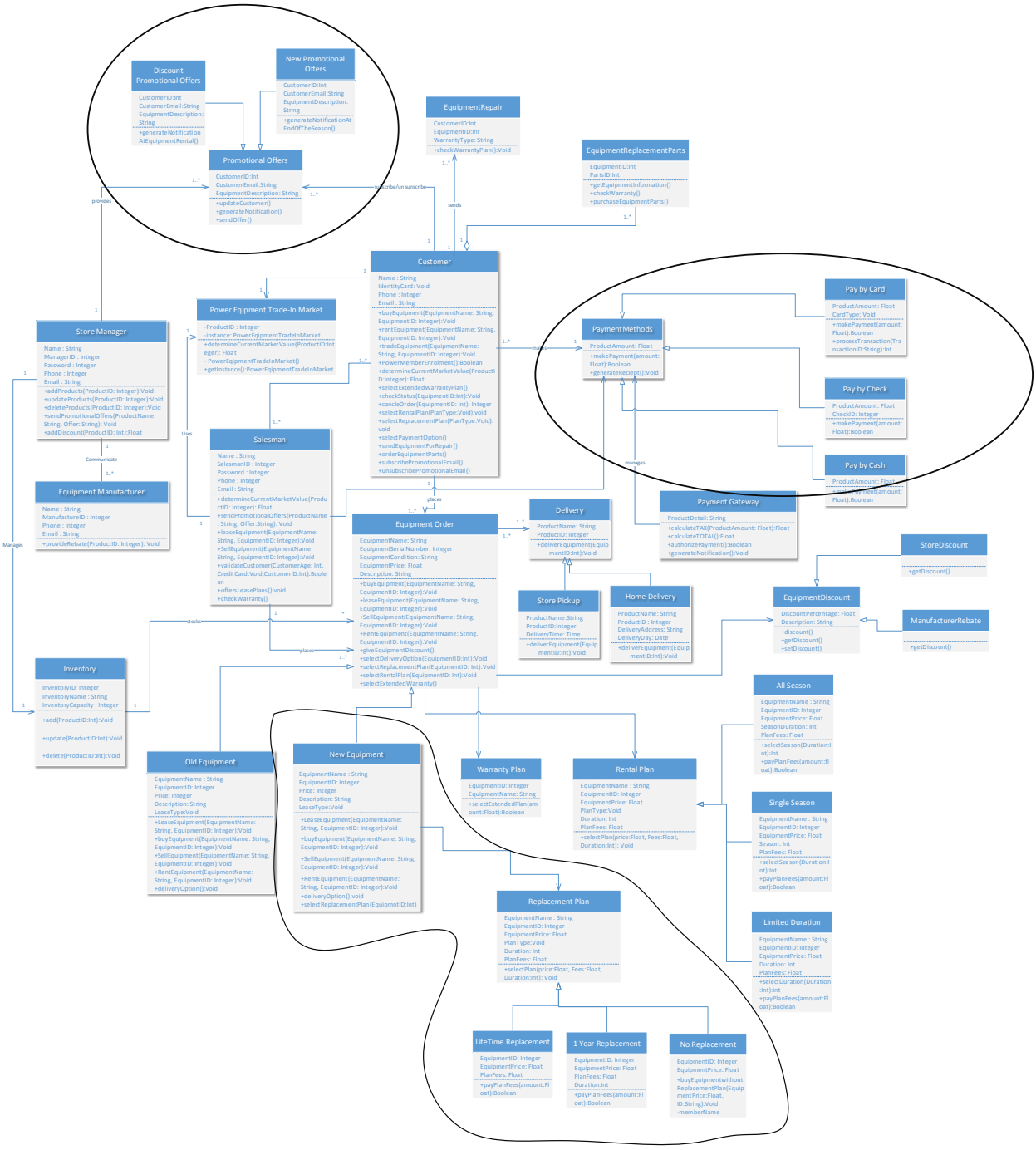
1. Design Model Class Diagram:

(Note: Classes involved in Abstract Factory pattern are not elaborate in this class diagram, I have expand them separately in diagram of Abstract Factory description)



- *Classes marked with* **BLACK border** *were involved in Assignment #3 design patterns.*
  - Observer Pattern: Used on E-mail Subscription module
  - Strategy Pattern: Used on Replacement Plan module
  - Factory Pattern: Used on Payment module
- ***Other Classes which are involve in NEW design patterns are explained separately in this documentation.***

2. List of Design patterns.

a. *NEW* Assignment 4 Patterns:

1. Abstract Factory Pattern: Used on Equipment & Payment module
2. Template Pattern: Used on Equipment Discount module
3. Singleton Pattern: Used on PowerEquipmentTradeInMarket module

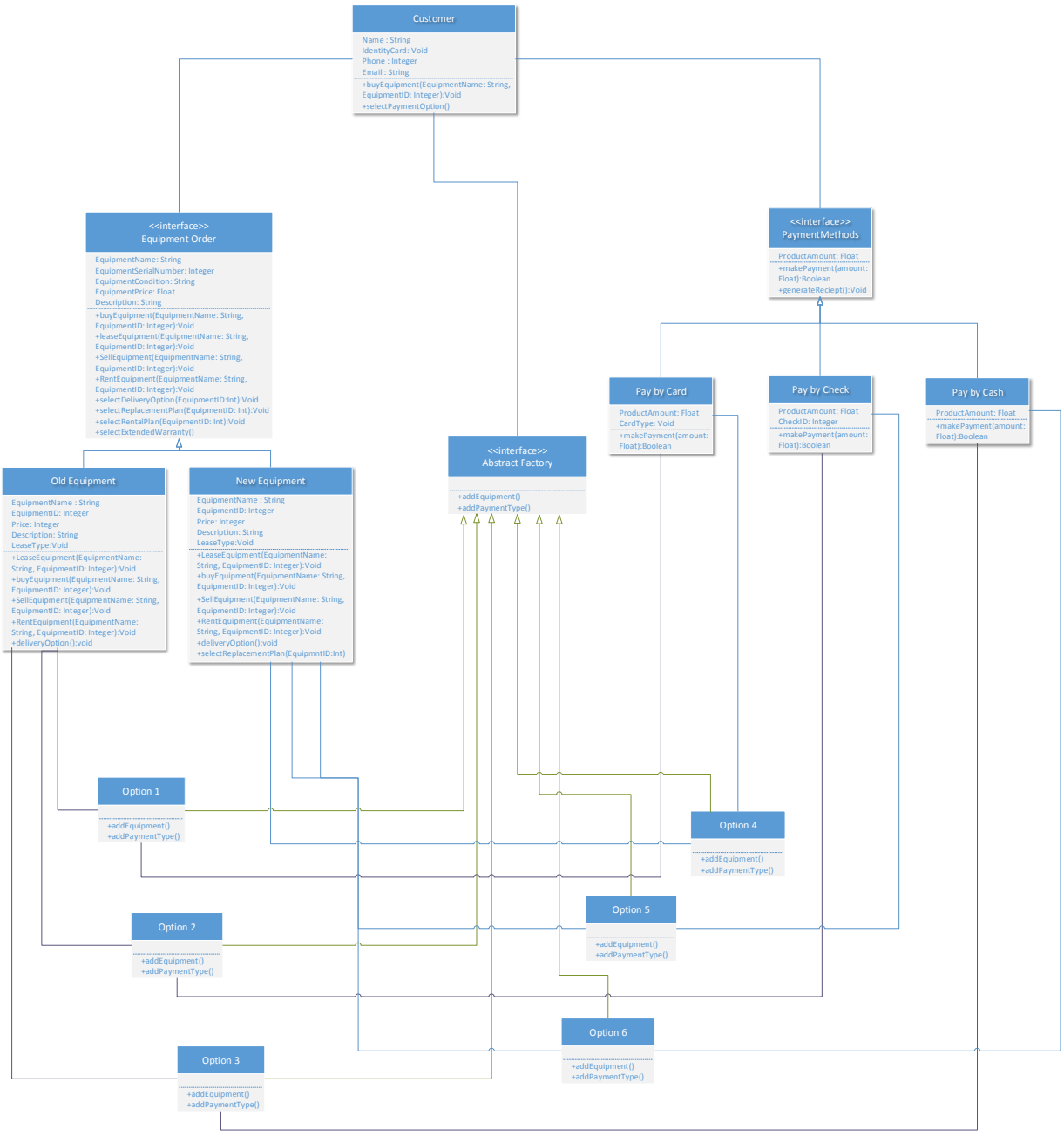
b. *OLD* Assignment 3 Patterns:

4. Observer Pattern: Used on E-mail Subscription module
5. Strategy Pattern: Used on Replacement Plan module
6. Factory Pattern: Used on Payment module

2. NEW Design patterns description.

a. Assignment 4 Patterns:

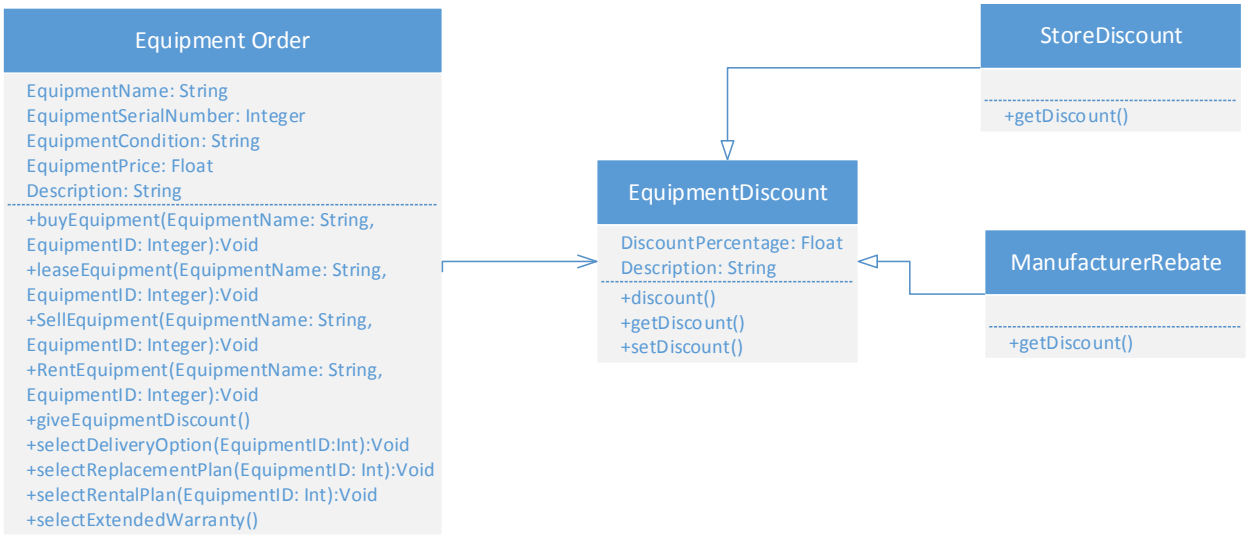
1. Abstract Factory Pattern: Used on Equipment & Payment module



- Abstract factory pattern is a hierarchy that encapsulates many possible "platforms", and the construction of a suite of "products". The purpose of the Abstract Factory is to provide an interface for creating families of related objects, without specifying concrete classes.
- In above diagram, Equipment Order & Payment Type are abstract products.
- Equipment Order declares an interface for a type of New & Old Equipment objects.
- Payment Method declares an interface for a type of cash, check & card payment type objects.
- Abstract Factory class declares an interface for operations that create abstract products.
- Options 1..6 classes implements operations to create concrete products.

- Here customer uses the interfaces declared by the Abstract Factory and AbstractProduct classes (Equipment Order & Payment Method).

2. Template Pattern: Used on Equipment Discount module



- Template pattern defines the steps of an algorithm and allows subclass to provide the implementation for one or more steps. The template method pattern is used to define the basic steps of an algorithm and allow the implementation of the individual steps to be changed.
- Here, Equipment Discount is an AbstractClass which defines abstract primitive operations that concrete subclasses Store Discount & manufacturer Rebate defines to implement steps of an algorithm.
- EquipmentDiscount class implements a template method which defines the skeleton of an algorithm. Template method getDiscount() calls primitive operations as well as operations defined in AbstractClass or those of other objects.
- StoreDiscount & ManufacturerRebate are ConcreteClass, which implements the primitive operations to carry out subclass-specific steps of the algorithm.
- When a concrete classes StoreDiscount & ManufacturerRebate called, the template method code will be executed from the base class while for each method used inside the template method will be called the implementation from the derived class.

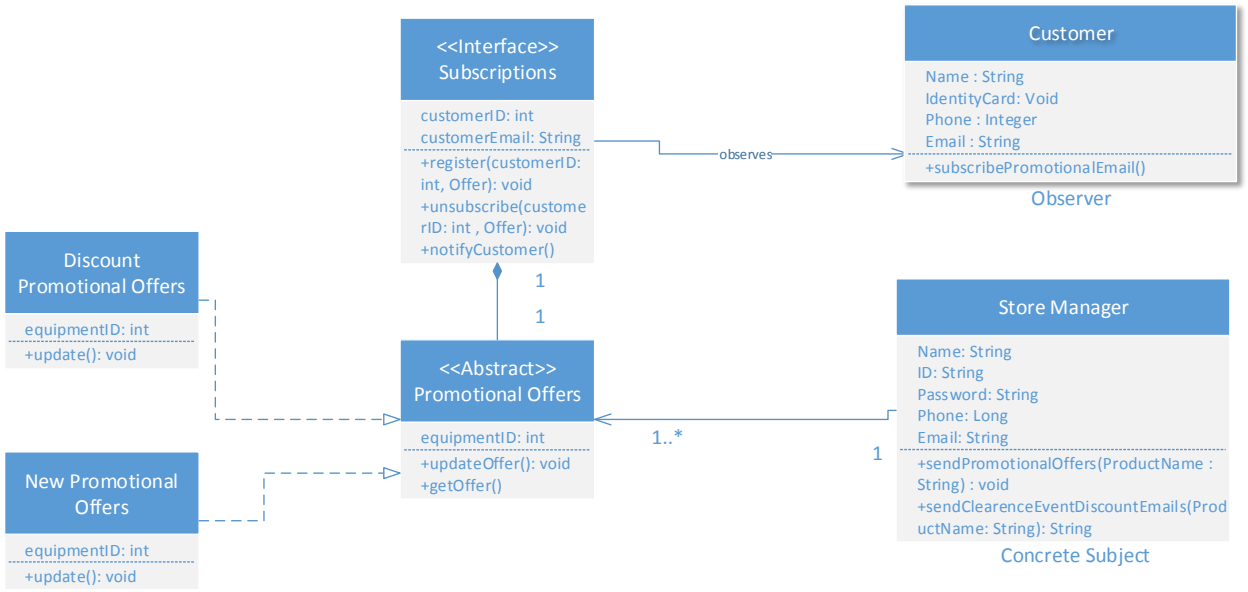
3. Singleton Pattern: Used on PowerEquipmentTradeInMarket module

- The singleton pattern is one of the simplest design patterns: it involves only one class which is responsible to instantiate itself, to make sure it creates not more than one instance.
- Here, “PowerEquipmentTradeInMarket” is defined as singleton design pattern.
- Above defined singleton class implementation involves a static member in the "PowerEquipmentTradeInMarket" class, a private constructor and a static public method that returns a reference to the static member.
- This Singleton Pattern defines a getInstance operation which exposes the unique instance which is accessed by the clients Customer & Salesman. getInstance() is responsible for creating its class unique instance in case it is not created yet and to return that instance.

3. *OLD* Design Patterns Description. (*This is just for information*)

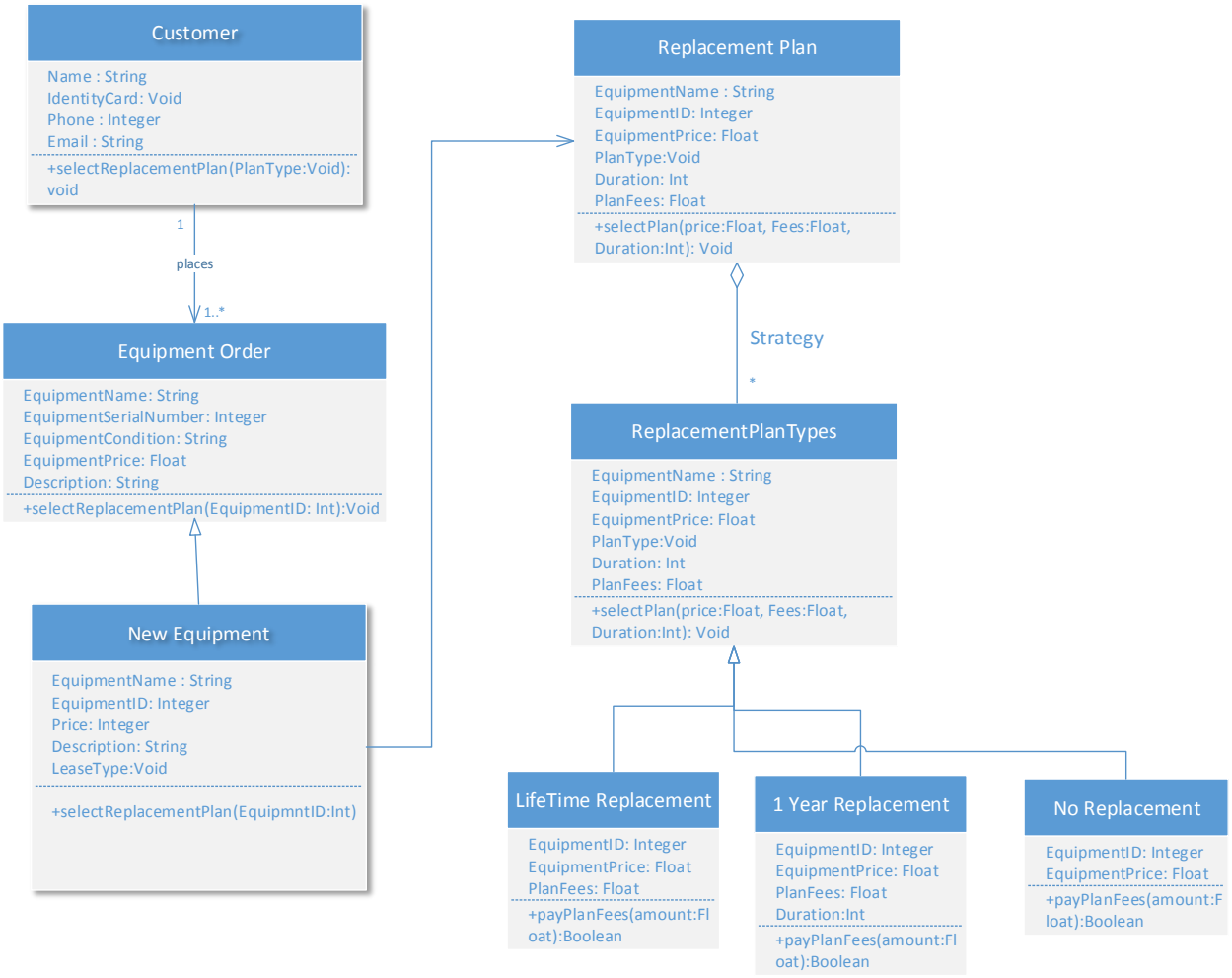
a. Assignment 3 Patterns:

4. Observer Pattern: Used on E-mail Subscription module



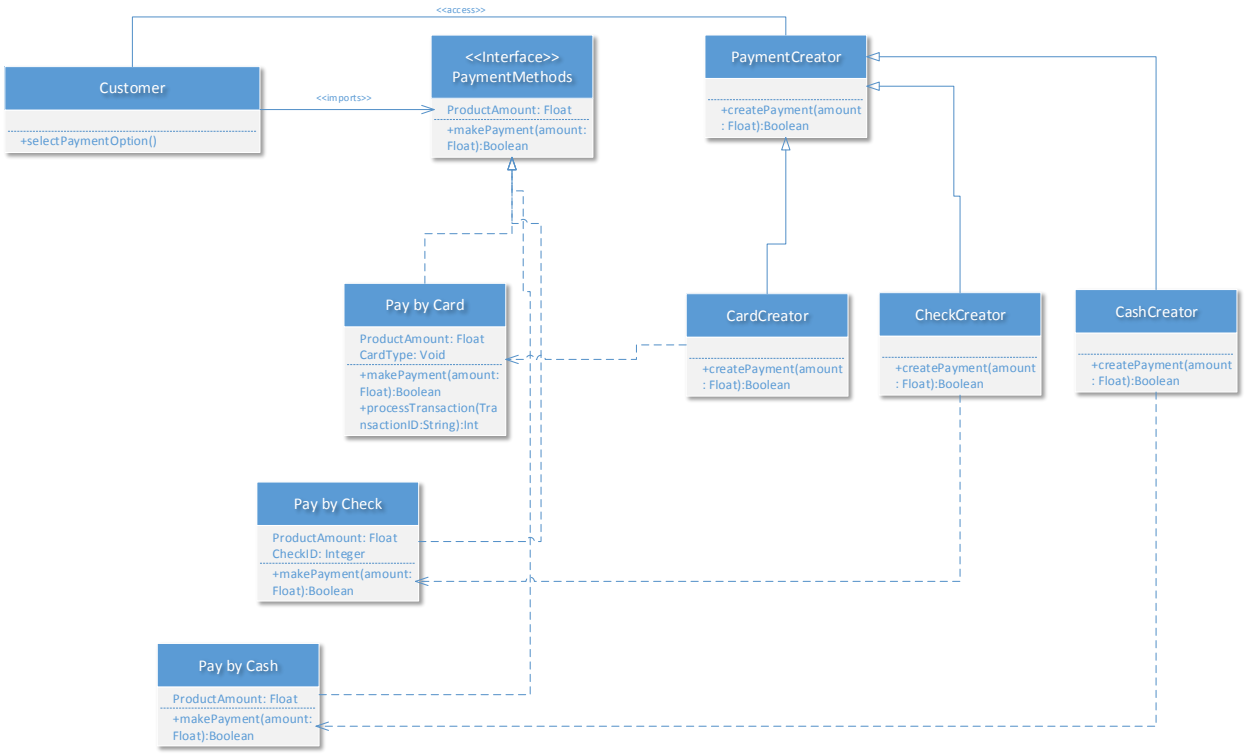
Here, Customer is the Observer, who use the Interface “Subscriptions” to get subscribe for promotional offers, Store manager generate promotional offers on different occasion, for ex. At the end of every season store manager sends 20% discount for clearance. Every time for storage space limitation threshold violate, storemanager sends 25% promotional offer to customer. Subscription Interface provide the customer to Register, unsubscribe from the email subscription. Promotional class provide updated offer to Interface.

5. Strategy Pattern: Used on Replacement Plan module



Here, any replacement plan can be selected from the three available plan which suited best to customer's needs while buying a NEW equipment. Strategy pattern is utilized on the above classes to show the proper strategy to choose appropriate plan for appropriate equipment & condition selected.

6. Factory Pattern: Used on Payment module



Here, Factory Pattern is implemented on Payment module class where we have various option to pay for the services we have used or items we have bought. All Payment methods have creator and they are connected to each of them. Payment Method is the Interface which provide various interface of payment type. Customer use this interface to select appropriate payment method suited best to his situation.