Table of Contents

[Primary problem 2](#_Toc399514006)

[Secondary problem 2](#_Toc399514007)

[Design Patterns used 2](#_Toc399514008)

[Final Compound Design 4](#_Toc399514009)

# Primary problem

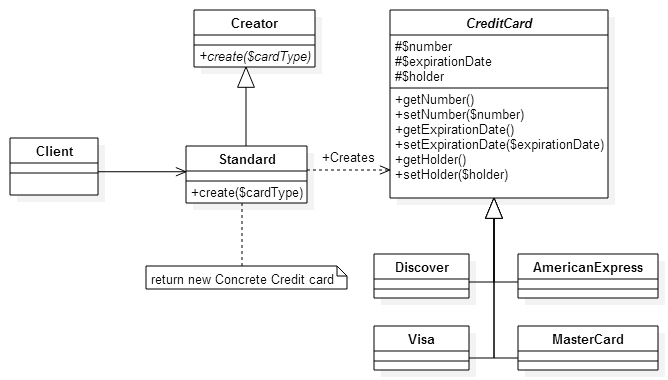
The primary problem is to create different instance of credit card based on the different type of card number type in an effective way where new card should be easily added or modified. While making the application highly maintainable.

# Secondary problem

The application should be loosely coupled and high cohesive, changes in the application should be localized and should not affect other components of the system

# Design Patterns used

1. Factory Method Pattern



Why and How

The factory method has been used to create the different credit card instances

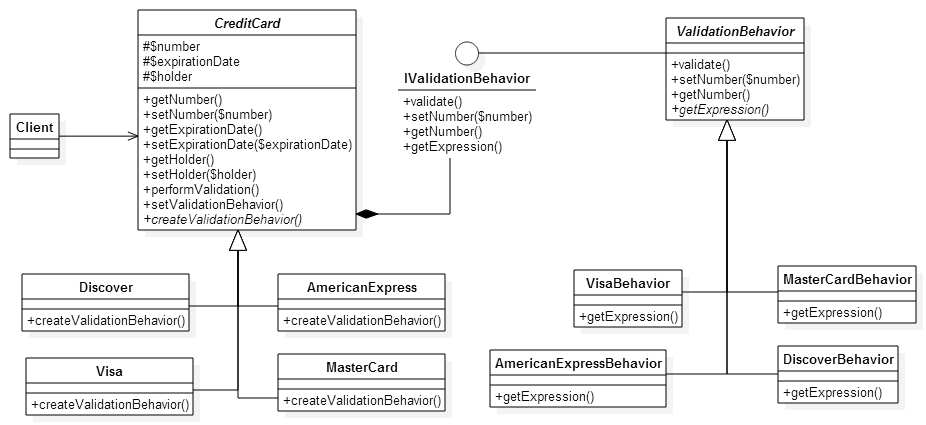
Consequences:

Pros:

1. Code is made more flexible and reusable since the card dependencies are removed from the client.
2. Credit card instances are lazily created.
3. Changes are localized.
4. Better maintained

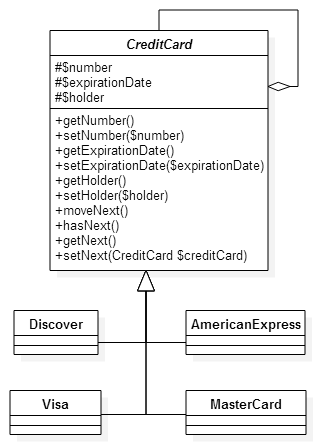
Cons:

1. May lead unnecessary subclassing when many cards will be added.
2. Strategy Pattern



* 1. The strategy pattern has been used to encapsulate the changing part, that is the different type of validations

1. Chains of Responsibility



* 1. Chain of responsibility has been used to chains the different credit card to be able to satisfied the request.

## Final Compound Design

