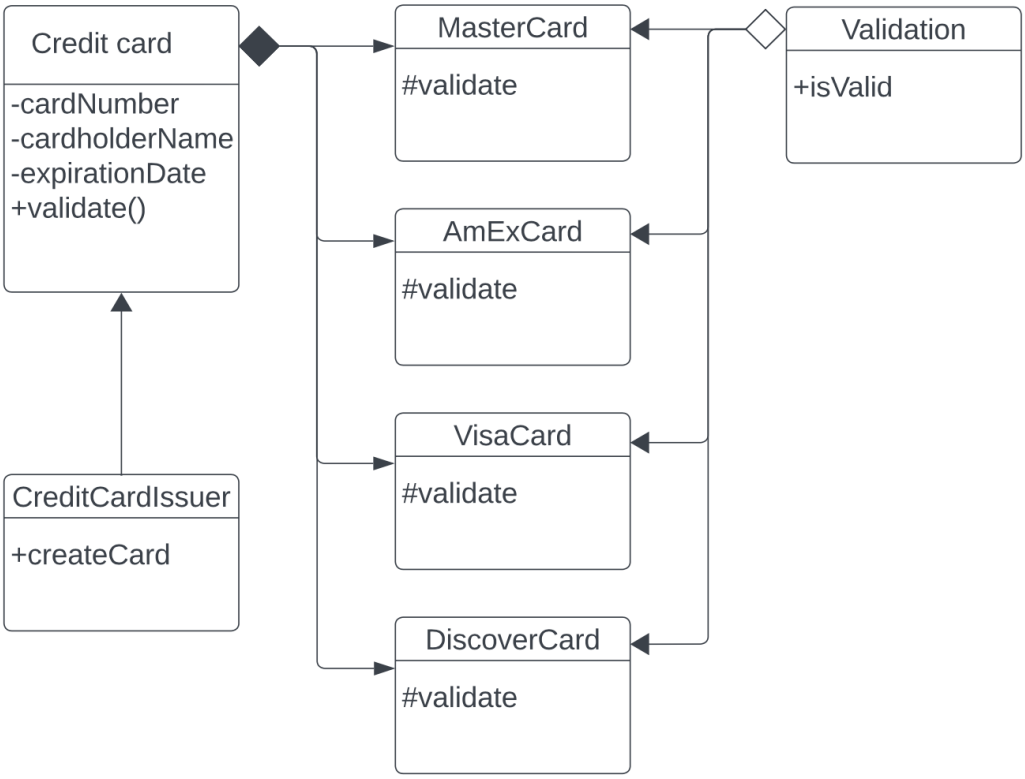
CMPE202 Individual Project
-Part I deliverables

Hao Lan
CMPE 202

Primary Problem Description:
The primary problem I aim to solve is developing a program that can accurately read and process credit card information. As for the project requirement, this program must have the availability to extract credit card details such as card number, expiration date, and cardholder's name from each record. After the card information is extracted, validate the credit card number based on the standard card issuer criteria. Finally, identify the issuer of the credit card. (Visa, MasterCard, American Express, or Discover).

Secondary Problem:
The secondary problem I aim to solve is ensuring the system is extensible to accommodate new credit card issuers in the future without big modifications to the existing codebase. Additionally, I am also thinking of a more efficient way to handle errors like unrecognized input. Current code only recognizes invalid card number digits input.

Design patterns:
In this design, I aim to address two primary challenges: determining the type of credit card based on its number and creating an instance of the corresponding credit card class.
Factory Design Pattern: The CreditCardIssuer class acts as the factory. It's responsible for creating instances of different CreditCard subclasses. When a card number is provided, the createCard method determines the type of credit card and instantiates the corresponding subclass.
Strategy Design Pattern: The Validation class and its interface isValid allow for the encapsulation of different validation algorithms. Each CreditCard subclass uses a specific validation strategy that is appropriate for its type. This design makes it easy to add new validation strategies for new card types without modifying existing code.



Consequences of design patterns:

Creational pattern (Factory pattern and singleton):

Pros:

1. Enhances system scalability by making it easier to introduce new card issuer types.
2. Reduces conditional logic in the application by moving the instantiation logic to a single location.

Cons:

1. This class type might be complicated to adjust when adding more card issuers and card types.
2. Might result in an extra layer of abstraction.

Behavioral pattern (Strategy pattern):

Pros:

1. Increases flexibility by allowing different algorithms to be swapped easily at runtime.
2. Makes it easier to scale and manage different validation rules for each card type.

Cons:

1. This might lead to an increased number of classes for each unique validation rule.