Part 1:

1. Primary problem is figuring out how to let my program decide different credit card types for the cardholder. There should be some validate method to go through all of the credit card types to check if the card is that type.

2. The secondary problem is figuring out how to instantiate that type of card decided by my program above. Once the card type is figured out, I should let my program return that card instance. There should be a class that is able to produce a CreditCard instance.

3. Based on the problems I described above, I definitely need a behavioral pattern which is able to decide the type of card during runtime, and also a creational pattern that is able to produce the card. For behavioral patterns, I choose chain of responsibility, because setting each validation as a handler, and connecting the handler to each other, then we have a chain of validation that a card must go through. This fulfilled the requirement I mentioned above.

   For card instance producers, I will have a factory method, this will create a corresponding CreditCard instance based on the validation type.

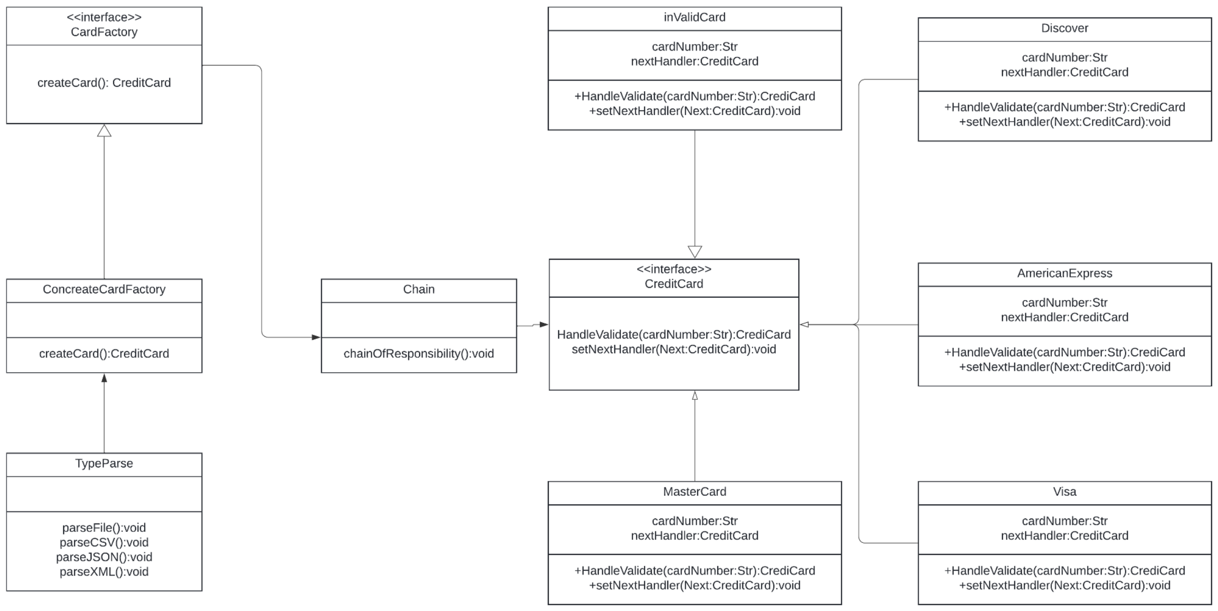
4. The consequence of chain of responsibility:
   a. allow dynamic decision during runtime
   b. Great design pattern, allow addition or removal of handlers easily without modifying code in alot of places
   c. Execution of handler in order, allow the card to go through the order as you planned.
   d. Process unknown and different cards

   The consequence of factory method:
   a. Encapsulate the object creation logic, making it easier to maintain or extend.
   b. Sticking to open/closed principle

Part2:

I will have another class TypeParse, this class will have 3 parsing methods, reading and writing to csv, json, and xml. I will have an extra method that client calls, this method will decide what parsing method to run based on the input file extension during runtime. The output should be the same as the input extension. In addition, it’s open for extension, for future file formats, I can just add a new method.

Class diagram



<<interface>>
CardFactory
createCard(): CreditCard
ConcreateCardFactory
createCard():CreditCard
TypeParse
parseFile():void
parseCSV():void
parseJSON():void
parseXML():void
Chain
chainOfResponsibility():void
inValidCard
cardNumber:Str
nextHandler:CreditCard
+HandleValidate(cardNumber:Str):CrediCard
+setNextHandler(Next:CreditCard):void
<<interface>>
CreditCard
HandleValidate(cardNumber:Str):CrediCard
setNextHandler(Next:CreditCard):void
MasterCard
cardNumber:Str
nextHandler:CreditCard
+HandleValidate(cardNumber:Str):CrediCard
+setNextHandler(Next:CreditCard):void
Discover
cardNumber:Str
nextHandler:CreditCard
+HandleValidate(cardNumber:Str):CrediCard
+setNextHandler(Next:CreditCard):void
AmericanExpress
cardNumber:Str
nextHandler:CreditCard
+HandleValidate(cardNumber:Str):CrediCard
+setNextHandler(Next:CreditCard):void
Visa
cardNumber:Str
nextHandler:CreditCard
+HandleValidate(cardNumber:Str):CrediCard
+setNextHandler(Next:CreditCard):void