CMPE202-Software Systems Engineering

Individual Project - Part 1

- Pankaj Patil

Problem Statement:

Credit card problem:

You have a **CSV** file that contains credit card records. Each record is on a line. It contains a field for the card number, the expiration date, and the name of the card holder. The fields are comma-separated. In your system you have the following class structure for the credit cards:

a class CreditCard,

classes VisaCC, MasterCC, AmExCC that are all subclasses of CreditCard,

you can assume more subclasses for other credit card types will be added later on.

You now have to design the method(s) (and maybe additional classes) that reads a record from the file, verifies that the credit card number is a possible account number, and creates an instance of the appropriate credit card class. What design patterns could you use for that?

Important details: Credit card numbers cannot exceed 19 digits, including a single check digit in the rightmost position. The exact algorithm for calculating the check digit as defined in ISO 2894/ANSI 4.13 is not important for this assignment. You can also determine the card issuer based on the credit card number:

MasterCard	First digit is a 5, second digit is in range 1 through 5 inclusive. Only valid length of number is 16 digits.
Visa	First digit is a 4. Length is either 13 or 16 digits.
AmericanExpress	First digit is a 3 and second digit a 4 or 7. Length is 15 digits.
Discover	First four digits are 6011. Length is 16 digits.

Deliverables:

Upload a PDF document containing the text and diagrams into Canvas. You can use Astah tool for the diagrams.

- Describe what is the primary problem you try to solve.
- Describe what are the secondary problems you try to solve (if there are any).
- Describe what design pattern(s) you use how (use plain text and diagrams).
- Describe the consequences of using this/these pattern(s).

Solution:

Q1. Describe what is the primary problem you try to solve:

a) The primary issue here is to resolve the card issuer based on the credit card number. As per the requirements we have four types of cards and we need to resolve card issuer based in the given criteria. More card issuers will be added later we need to take that into account while designing.

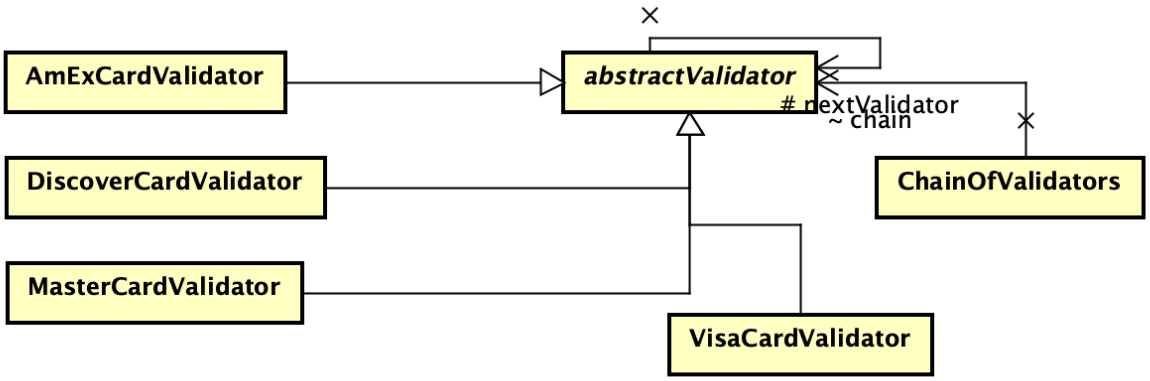
Q2. Describe what are the secondary problems you tried to solve:

Secondary Problems:

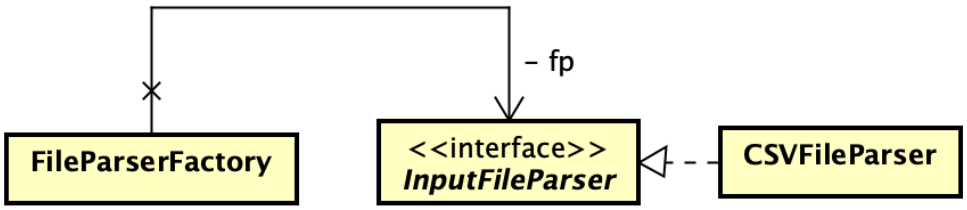
a) Read and parse the input CSV file. CSV file contains three fields (card number, the expiration date, and the name of the cardholder)
b) How to create an appropriate instance of credit card class based on the card issuer (Card Type). Currently four types of card issuers are given

Q3. Describe what design pattern(s) you use how (use plain text and diagrams).

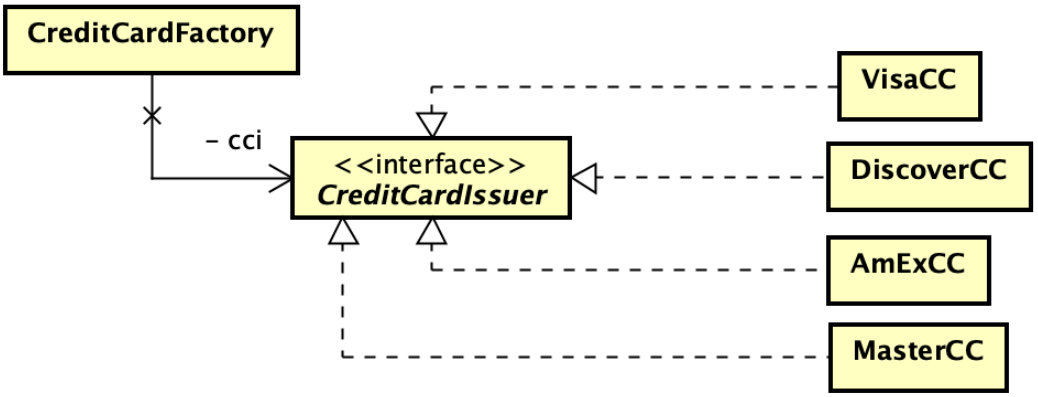
a) To resolve the card issuer based on given criteria, I am planning to use the "Chain of Responsibility Pattern". The "ChainOfValidators" class will create a chain(set hierarchy of validators). Each validator will verify the card number and if it doesn't fit in his criteria then it will pass to another validator. If card no matches with any validator then that validator will return card issuer name and no further validators will be called.



b) Currently, only 1 type of input file is given but, in the future, that could change, and we might get data in different formats. To support different types of input file I will use the factory method pattern to create an appropriate class to handle the parsing of the file and returning the list of type 'CreditCard'(POJO) which will contain three fields as 'cardNo', 'expirationDate', and 'cardHolderName'. With this design pattern in the future if required we can add a new class and the main object code will not change. I will derive file extension from "filepath" and use that to create an instance of an appropriate file parser class.

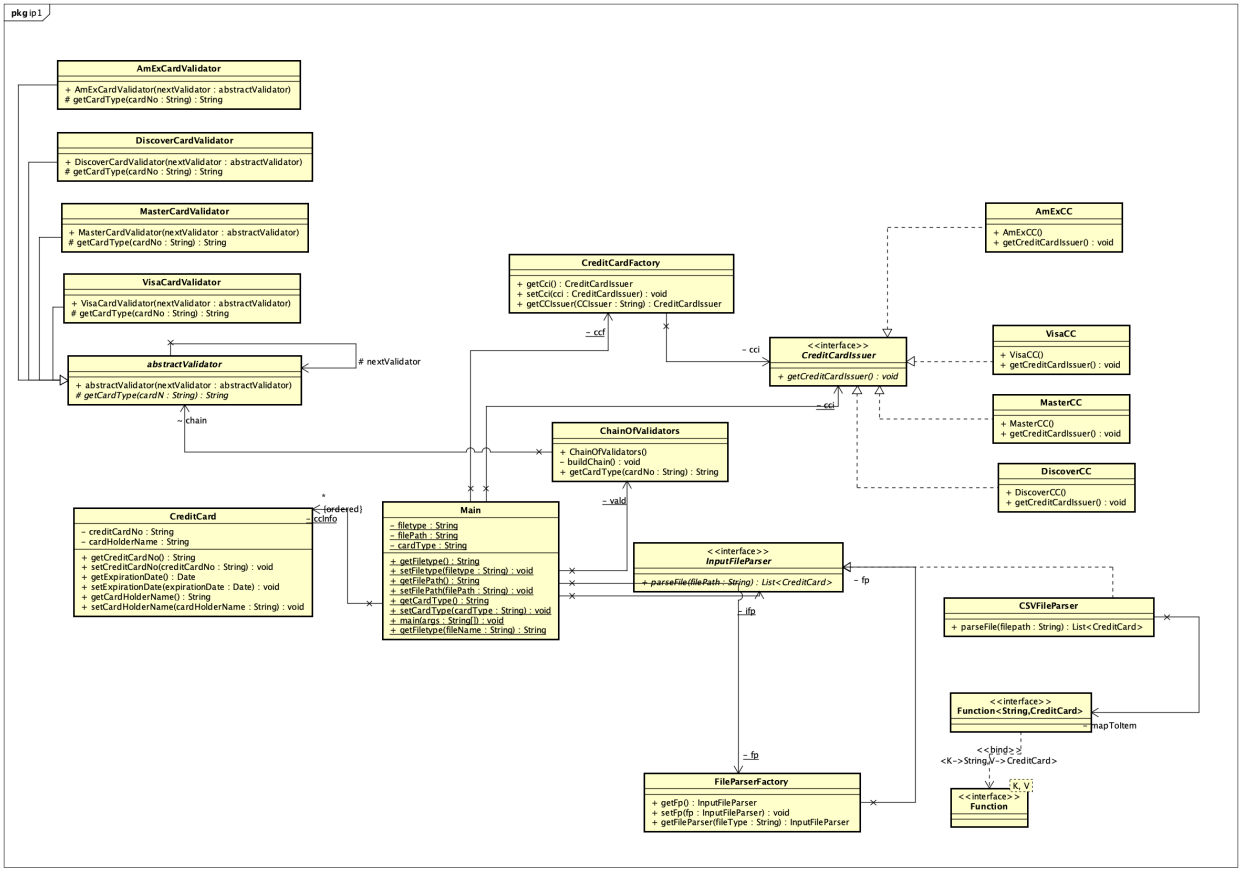


c) Lastly depending on the card issuer, we need to create the appropriate object. to create objects based on the card issuer, I am planning to use the factory method pattern for these object creations. I will use “CreditCardFactory” to create an appropriate instance of the class by passing the card issuer name. We will get the card issuer with solution to primary problem(Find Card Issuer based on credit card number)



Overall design:

Parse CSV → Create list of “CreditCard” Object → Validate each record to determine Card Issuer → Generate appropriate instance



Q4. Describe the consequences of using this/these pattern(s).

Factory method pattern has the following consequences:

a) Factory methods eliminate the need to bind application-specific classes into your code. The code only deals with the Product interface. This makes adding new classes and modifying existing once flexible and without causing the main class to recompile (Loose coupling)
b) With the Factory method pattern, we can defer the creation of class to subclass. This is an advantage and a potential disadvantage. Subclassing is fine when the class has those classes anyways (subclass and creator class), otherwise client now must deal with another point for potential change.

c) Creating objects inside a class with a factory method is always more flexible than creating an object directly. Factory Method gives subclasses a hook for providing an extended version of an object

Chain of Responsibility has the following consequences:

a) With Chain of Responsibility pattern main object needs not worry from knowing which another object handles a request as long as the request is handled appropriately. With this, we can decouple the receiver and the sender simplifying object interactions.
b) Chain of Responsibility gives you added flexibility in distributing responsibilities among objects. This pattern makes it easier to add new and modify existing conditions.
c) Since a request has no explicit receiver, there's no guarantee it'll be handled. If the chain is not configured properly then the request might go unhandled.

Reference:

http://klevas.mif.vu.lt/~plukas/resources/DPBook/excerpts/factmeth/conseq.htm
http://klevas.mif.vu.lt/~plukas/resources/DPBook/excerpts/chain/conseq.htm
https://www.tutorialspoint.com/design_pattern

CMPE202-Software Systems Engineering

Individual Project - Part 2 and 3

- Pankaj Patil

Problem Statement:

Part2: 15 points (Design only)

Continue with the design from Part 1 and extend it to parse different input file formats (json, xml, csv) and detect the type of credit card and then output (in the same format as input) - into another file - with each line showing the card number, type of card (if a valid card number) and an error (if the card number is not valid). The design should accommodate newer file formats for the future.

Part 3: 60 points

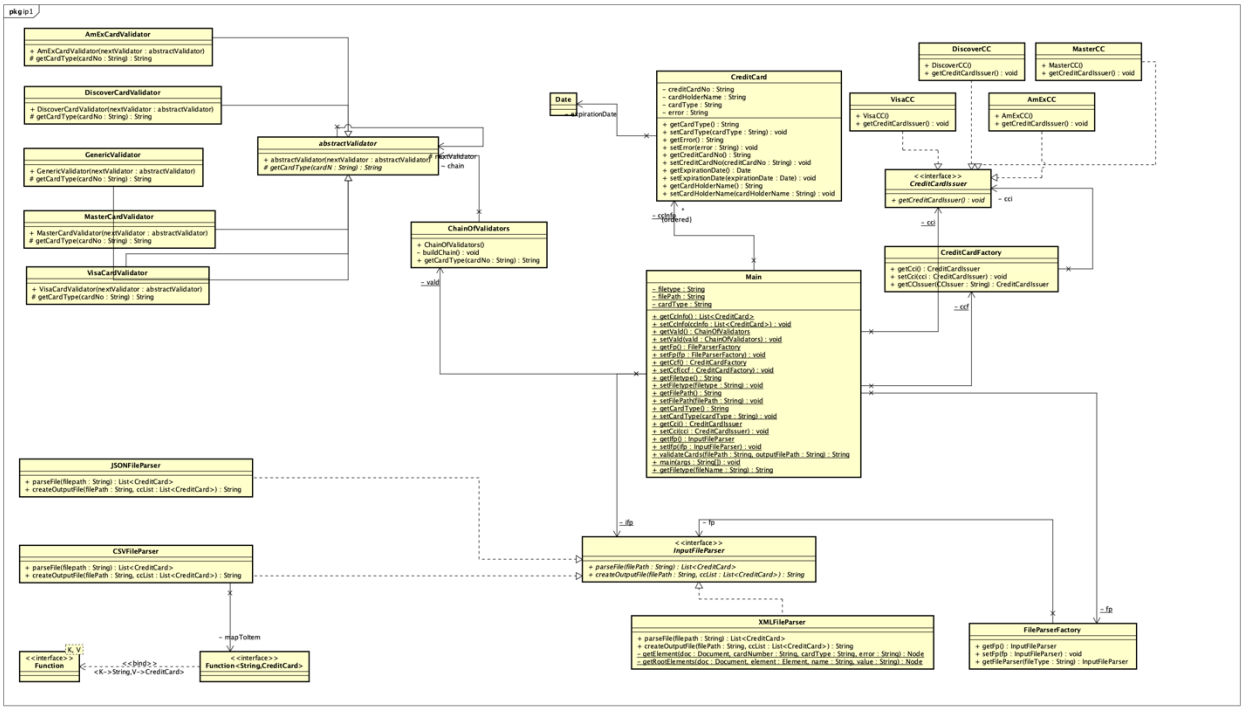
Implement an application (Java code and JUnit tests) for Part 1 and Part2 - that accepts input file name and output file name and writes an output file in the same format as the input (CSV or JSON or XML). Output should contain the details specified in Part 2.

Submit design and code in the github repo (classroom invite will be sent by grader)

Grader will provide sample JUnit formats and input file formats

Solution:

Final Solution Class Diagram



Code Repository:

https://github.com/gopinathsjsu/individual-project-pankajhpatil.git

Reference:

https://www.tutorialspoint.com/design_pattern
https://docs.oracle.com/en/java/javase/index.html
https://junit.org/junit5/docs/current/user-guide/
https://mkyong.com/java/how-to-read-xml-file-in-java-dom-parser/